FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Artistic Style Transfer for Textured 3D Models

Inês Filipa Nunes Teixeira

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Maria Teresa Andrade

Second Supervisor: Peter Eisert

July 30, 2017

Abstract

This is not an artistic document. Yet, it is not exclusively technical, either.

Style Transfer is an image transformation task that relies on generating an image with a balanced combination of content and style from different sources. This dissertation focuses on applying this same principle to three-dimensional textured models, providing a solution for transferring any style from an input image to a texture map. Style can be considered as a matter of colour, texture, light, shade or any distinctive way that characterises, in this case, an image. Actually, the definition of style varies with the context - for the aims of this dissertation, style is intended as a combination of colour and texture.

The solution presented in this dissertation has several inputs, namely the content image, the style image, the object file, a mask from the texture map and a noise image. The content image is the original texture map of the three-dimensional model to be modified; the style image represents the texture and the colour to be applied to the model to produce the desired output; the object file describes the three-dimensional model; the mask from the texture map enables the recognition of texture islands borders of the same model; and a random noise image, which will be subjected to an iterative process of fine detailed adjustments according to the characteristics of the input model and of the input style image, converging to the output with the desired style transfer. The proposed solution uses a Convolutional Neural Network trained on object recognition for approximating in different ways the feature responses of the noise image to the content and style images. In order to ensure a smooth appliance of style to the texture map, in a way that when it is mapped into the 3D model, the result is visually appealing, some more considerations have been taken into account, namely by creating a new constraint to ensure smooth transitions between texture islands. The developed method was effective and it does not depend on any specific software, as the algorithm is applied between images before being visualised in three-dimensions. The algorithm supporting the method was written in the Python language, and has used the framework Caffe for the Convolutional Neural Network.

This document describes the project developed in the scope of the master's thesis on Electrical and Computers Engineering from the Faculdade de Engenharia da Universidade do Porto(Faculty of Engineering of Porto University). This dissertation was developed in the Heinrich-Hertz-Institute (HHI), Fraunhofer, Berlin.

Resumo

Este não é um documento artístico. No entanto, também não é apenas técnico.

A transferência de estilos é um método de transformação visual que se baseia em gerar uma imagem através da combinação equilibrada entre conteúdo e estilo, de origens diferentes. Esta dissertação concentra-se na aplicação deste mesmo princípio a modelos tridimensionais texturizados, desenvolvendo uma solução para transferência de estilos de uma imagem para um mapa de texturas. Estilo pode ser definido como cor, textura, luz, sombra ou qualquer forma distinta que caracterize, neste caso, uma imagem. Assim, poderia dizer-se que a definição de estilo depende do contexto. Nesta dissertação, estilo refere-se a uma combinação de cor e textura. A aplicação do método requere vários dados de entrada (inputs), nomeadamente, a imagem de conteúdo, a imagem de estilo, o arquivo-objeto, uma máscara, o mapa de textura e uma imagem do ruído. A imagem de conteúdo é o mapa de textura original do modelo; a imagem de estilo representa a textura e a cor a aplicar nos resultados (outputs); o arquivo-objeto descreve o modelo tridimensional; a máscara de textura permite o reconhecimento de fronteiras de ilhas de textura do mesmo modelo; e a imagem do ruído, que será ligeiramente ajustada repetidamente até a convergência final.

O método proposto utiliza uma rede neuronal convolucional capaz de reconhecer, de diferentes maneiras, as caraterísticas das respostas da imagem de ruído às imagens de conteúdo e estilo. Para garantir a aplicação suave do estilo ao mapa de textura, algumas considerações adicionais foram implementadas, nomeadamente adicionando uma restrição espacial ao sistema, para garantir uma evolução suave da textura quando o resultado é mapeado no modelo tridimensional. O método desenvolvido não depende de nenhum software específico, uma vez que a adaptação é aplicada entre as imagens e só posteriormente pode ser visualizada em três dimensões. O algoritmo foi escrito na linguagem Python, e usou o framework Caffe para a rede neural convolucional.

Este documento descreve o projeto desenvolvido no âmbito da dissertação de mestrado em Elétrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto. A dissertação foi desenvolvida no Instituto Heinrich-Hertz (HHI), Fraunhofer de Berlim.

Acknowledgements

I would first like to thank my thesis advisors, Prof. Maria Teresa Andrade from the Faculdade de Engenharia da Universidade do Porto (Faculty of Engineering at the University of Porto, Portugal) and Prof. Peter Eisert from the Heinrich-Hertz-Institute (HHI), Fraunhofer, Berlin, Germany for the opportunity I have been given for developing this dissertation in this thrilling subject and at this respectful world class research institution. They have steered me during the exhausting months I needed to complete my work and definitely contributed to the success of it by constantly helping me whenever I needed guidance or assistance.

I would also like to thank the other researchers for their availability helping me with their expertise whenever I ran into a trouble spot or had a question about my research. However, they consistently allowed this dissertation to be my own work, and I sincerely thank them for that. Special thanks are due to Lisa Kausch for helping me whenever I needed.

Thanks are also due to friends and university mates who were involved in the reading and validation of the thesis: [Vicente, Luís, Ivo, Lúcia], for their support [Sara, Ana, Eduardo, Manuel, Daniela, Ângela, Ricardo, Guilherme, André] and all the others for their friendship during my academic journey.

Finally, I would like to acknowledge my family for supporting me during my stay in Berlin and for their love. A special thanks are due to my father, for his incredible patience, commitment and availability, to my mother, for giving me all the support I needed always one call away, and for my two best tenants, my brother and sister, for bringing me joy and excitement anywhere and at any time.

Inês Teixeira

"Mi svegliai, solo per vedere che il resto del mondo era ancora addormentato "

Leonardo Da Vinci

Contents

List of Figures

List of Tables

Abbreviations and Acronyms

2D	Two-dimensional
3D	Three-dimensional
AI	Artificial Intelligence
AR	Artistic Rendering
CNN	Convolutional Neural Network
Conv	Convolutional Layer
IB-AR	Image-Based Artistic Rendering
NPR	Non-Photorealistic Rendering
ReLU	Rectified Linear Unit
RGB	Red Green Blue
VGG	Visual Geometry Group

Chapter 1

Introduction

The focus of this dissertation is on the development of a an approach to apply artistic styles to textured three-dimensional models. Three-dimensional (3D) computer graphics are graphics that use 3D representation. 3D models are mathematical representations of any surface of an object in three dimensions. Stating that a 3D model is textured merely means there is a texture mapped to this model, producing a more visually appealing result, either to render a more realistic representation or a more artistic one.

Having any textured 3D model, the proposed method allows to apply of a 2D style representation (e.g. a painting) in its texture map. This requires the identification of style by the computer, producing a separation between content and style. The ability to look at any image and identifying the style associated to it is a cognitive function, i.e. it requires intelligence. For a computer to be able to accomplish this goal, it needs to analyse the image data, extracting relevant characteristics from it and process that information, performing reasoning operations. This may need to be conducted repeatedly, to fine-tune earlier judgements. Part of these tasks can only be performed through the use of Artificial Intelligence (AI) techniques, in particular, Machine Learning[1].

In 1959, Arthur Samuel, an American computer scientist pioneer in the field of computer gaming and artificial intelligence, defined machine learning as the "field of study that gives computers the ability to learn without being explicitly programmed" [Munoz, 2014]. In this dissertation, the style transfer technique evokes a type of Artificial Neural Network [2], a Deep Learning [3] Convolutional Neural Network (CNN), in which the connectivity pattern between the more basic units (i.e. neurons) is inspired by the organisation of an animal neural system, e.g. the visual cortex[4].

[1] Machine Learning is a sub-field of computer science and is one application of AI that focuses on the ability of learning by a computer.

[2] Artificial Neural Networks are computational models used in Machine Learning, based on a large collection of connected simple units called artificial neurons. These have various applications, namely computer vision and speech recognition.

[3] Deep learning refers to Artificial Neural Networks that are composed by many layers

[4] Consult appendix A for more detail

1.1 Relevance

Computer graphics is an exciting and growing field of computer science [Foley *et al.* , 1994], [Hearn & Baker, 1994], and neural networks are currently providing the best possible solutions for image recognition [Nielsen, 2017]. Putting together and connecting computer science to the artistic usability, it is possible to create a system that combines science and art. This system could be seen as a new form of art, as a new application for digital art or as a tool to help creators create, i.e. help artists to produce new works of art.

The importance of this kind of systems is related to the change we are living nowadays: one of the greatest shifts in human communication ever [Vaynerchuk, 2016]. Finding solutions able to participate in this change, by generating things never seen before, is playing a role in this shift and collaborating in the wave of innovation.

Related to this, Facebook highlighted three core pillars to a "ten years from now" challenge. One of them is solving problems in Artificial Intelligence (AI) systems that help manage the huge amount of online information. However, AI and, more specifically, neural networks were designed in light of the human brain. As such, they are not only fact tools, but they can also be creative, and so, help us create as well [Schroepfer, 2016b]. An example of such kind of use of these tools are the psychedelic and abstract images generated by the deep learning "dream" systems [Titcomb, 2015], [Tyka, 2015a], [Tyka, 2015b], [Tyka, 2015c].

There is no doubt that deep learning is playing an important role nowadays, because it is a powerful technology and because deep learning community has structured itself to facilitate innovation very quickly. First, by becoming much better at exploiting computational resources, with high quality open source code libraries, and second, by becoming good at discussing and sharing information with everyone about how to do deep learning and what are the recent breakthroughs Larochelle [2016]. It is interesting to highlight a resemblance with the evolution of the Internet, comparing to its rapid growth, as it did not have a single inventor, but it evolved quickly over time [History, 2010]. Neural networks were designed more than 30 years ago, but they are only having this incredible breakthrough now because powerful processing machines were not so easily available and there was not such a big amount of information available online, which is required to train the networks [Vigoda, 2016]. Going back in history, Frank Rosenblatt could probably be considered the first trace of Neural Networks, in 1957. The Mark 1 perceptron machine was the first implementation of the perceptron algorithm, describing a small unit that processed a binary step function, which defined an activation function with learning rules considering inputs and weights. It was a hardware implementation that could recognise letters of the alphabet. The first multilayer system was designed by Widrow and Hoff, in 1960, which was also a hardware implementation of a perceptron network. The backpropagation technique was only referred by Rumelhart, Hinton and Williams in 1986, in a multilayer system with backpropagation to calculate the gradient descent, but the training of the network was not producing the expected results and so there was not much research in this field for some time. Hinton & Salakhutdinov, in 2016, proposed a new system of a multilayer network that had better results in the training process, but this was achieved

by dividing it in two stages: firstly training layer by layer (a step wise training through the layers) and secondly using backpropagation at a single pass for tuning. Unfortunately, this strategy did not produce good results with deeper and larger networks. Large improvements were denoted around 2010, applying these systems for tasks related to speech recognition [Dahl *et al.* , 2012] and image classification [Krizhevsky *et al.* , 2012], where the results were very promising and since then there was a boost in research on this field [Karpathy, 2016].

Consequently the evolution of these systems and to the utmost need for creativity, a new idea of style transfer is born [Gatys *et al.* , 2015]. Style transfer is a method of transferring the style of one image to another, maintaining the content of the second. So, in a shortened form, finding a way to separate style from content and create a new image combining these two, from different sources. This method is motivated by the observation that the artistic style of a painting may be interpreted as a visual texture [Gatys *et al.* , 2015], [Dumoulin *et al.* , 2016]. A visual texture is conjectured to be spatially homogeneous and consist of repeated structural motives whose minimal sufficient statistics are captured by lower order statistical measurements [Dumoulin *et al.* , 2016].

Despite considering the painting's style as a visual texture, it is important to retain that the style of a painting does not uniquely characterise an artist, as the style of an artistic movement does not translate entirely in a single painting [Ikuta *et al.* , 2016]. As such, the purpose and meaning of the algorithm developed by Gatys et al. [Gatys *et al.* , 2015] is to transfer a style from one painting, created by one artist, to an image of choice. Obviously, the result of this is not certainly a possible painting by the same artist, but a visually appealing composition presenting characteristics similar to those that can be found in an artefact created by the artist. This can be of much use for helping anyone to be creative and could serve as a baseline for a new creation. The areas of application of these systems are mainly social networks [Moiseenkov, 2016], [Schroepfer, 2016a], games [Wilmot *et al.* , 2017] and cinema [Joshi *et al.* , 2017]. [Joshi *et al.* , 2017]. However, the more minds trying to use this technology, the more ideas of new applications will arise.

The main goal of this dissertation is to investigate the possibility of developing an efficient solution for transferring style to 3D models, based on the work developed by Gatys et al. [Gatys *et al.* , 2015]. More specifically, it is aimed to adapt the deep learning algorithm from a pure 2D operation into a three dimensional world, identifying the main challenges and proposing solutions to overcome those challenges.

1.2 Background to the addressed subject

History tells stories of moments when the feeling of challenge and the need for change resulted in big shifts on reality. Not only concerning science and its findings, but also in artistic movements that marked our past and taught humankind what is known today. These were remarkable changes in the artistic world, namely appearing to oppose the existing principles of the time. After the Middle Age, the first artistic movement was the Renaissance, which literally means rebirth, by the fourteenth century. The plight of Renaissance was to move away from the religion-dominated themes and focus in individual expression. Reacting to the harmonious ideals of this period came

Mannerism, seeking instability and restlessness. This was replaced by the Baroque in 1600, which opposed the infringement of rules of the previous movement. Baroque was encouraged as a return to tradition and spirituality, and so it is described as complex and emotional, sometimes denoting an exaggeration to produce drama. This evolved to the Neoclassicism, and was opposed almost simultaneously by the Romanticism, with its emphasis on emotion, individualism and theatrical drama. This fight between innovation and tradition, changing and reviving the past, was evident during these centuries and led to the Modern Art Era, a big breakthrough in the history of art [Artindustri, 2013].

"Art is never finished, only abandoned"

Leonardo Da Vinci

In the first decades of the nineteenth century, the Modern Art was born with the dawn of the Industrial Revolution [theartstory, 2009]. By this time there was already a technology that recorded an image through the action of light [Andy Grundberg, 2017]. As this technique of photography improved, it became increasingly more accessible to the general public, withdrawing the purpose of using painting as a way of documenting [theartstory, 2009]. For this reason, photography was somehow the stimulation for artistic diversity in the nineteenth century [Kyprianidis *et al.* , 2013], and by this time a new artistic movement had emerged, contradicting the idealisation, fantasy and physical beauty ideals of previous artistic times [theartstory, 2009]. And so, artists found new ways of expression.

"I paint objects as I think them, not as I see them"

Pablo Picasso

The Modern Art Era is believed to have begun with the Realism movement, that came against the formulas of the Neoclassicism and the theatrical drama of Romanticism. Realism sought to represent realistic scenes as they were, sometimes involving an implicit moral message, without avoiding the depiction of the ugly. This served as a statement, provoking a series of artistic movements, without a strict temporal sequence. Impressionism marked a momentous break from tradition, because it was a sudden change on the look of paintings. It was based on perception as a glimpse of a scene. After this came the Post-Impressionism, Fauvism, Expressionism, Symbolism, Cubism, Futurism, Dadaism, Surrealism, Abstractionism, Pop Art and many others until today's Postmodernism [Artindustri, 2013].

As photography stimulated this artistic diversity in the Modern Art Era, so did photorealism techniques in computer graphics motivate alternative techniques to create non-photorealistic styles [Kyprianidis *et al.* , 2013]. Non-photorealistic renderings (NPR) is an area of computer graphics dedicated to enable a wide range of styles for digital art, as it "gives freedom to encode an impression of the scenes rather than being forced to follow physical constraints" [Geng, 2011]. Applying a Style Transfer technique is a new opportunity to implement an art form in computer graphics, somehow following NPR principles [Yongcheng Jing & Song, 2017], and a significant

contribution for nowadays' need for creativity. In fact, according to the Adobe study [Jana, n.d.], "Creativity is regarded as one of the top three personality traits most important to career success". Also, in a IBM global study, CEO's picked creativity as one of the most important leadership qualities [Tomasco, n.d.].

Most of the effort of applying the Style Transfer algorithm [Gatys *et al.* , 2015] has been placed in 2D [Selim *et al.* , 2016], [Ikuta *et al.* , 2016], [Wilmot *et al.* , 2017] and video [Ruder *et al.* , 2016a], whilst 3D style transfer has deserved much less attention [Fišer *et al.* , 2016], [Ma *et al.* , 2014]. This is partially because the topic is rather recent and also due to the requirements in computational resources for applying such processes into 3D models. Accordingly, 3D style transfer remains a challenge and in fact, a very few references on work carried out in this topic may be found in the literature.

Although most of the effort of Style Transfer has been placed in 2D, developing 3D style transfer could be beneficial for a wide range of applications, namely generating artistic models with a chosen style, recreating a virtual environment with editable models[5]. This meets the goal of using style transfer as a mean to help create new things. Another example of this is the recent project developed by Artomatix [Dromey, 2017], [Artomatix, 2014] which consists on a creative artificial intelligence technology that helps a digital artist create a variety of different characters based on the same general characteristics.

Also, from a scientific perspective, this could be a first step to the creation of documenting models in a visually appealing way, to help users to learn to recognise patterns and details. For instance, considering that a lot of information on science is available online, if one needs to learn about a topic, one must read articles and blogs on the matter. With a virtual system, it would be possible to use the knowledge available on the enormous database that is the internet, and create a simulator for human diseases, plant diseases, mechanical problems on machines and materials. Having a rash, a patient can identity the name of the disease online by observing the model and contact the correct specialised doctor on the matter. Having a wine, an inexperienced agriculture can identify the disease on a leave and treat the plant accordingly. This dissertation aims at exploring a first approach in applying an AI technique to 3D models appearance by presenting a comprehensive solution for problems detected in 3D artistic style transfer.

1.3 Objectives and Research Question

The main goal of this dissertation is to transfer an artistic style from a 2D image, and so not requiring any artistic knowledge, to a textured 3D model and identifying the problems that may appear in this process. In other words, the scope of this dissertation relies on the question: "Is it possible to transfer a style from a 2D image into a 3D model, by means of a Deep Style Transfer algorithm?". To answer this, it is important to guarantee the following conditions: 1) Is it possible

[5] What could recall to these well known projects [Bear, 2001], [Parade, 1999]

to solve the discontinuity problems[6] that appear between texture islands in the model? and 2) Will this implementation produce visually appealing results?

Later on, answers will be presented to these questions alongside with the results obtained by using the developed solution.

1.4 Methodology

Few research on transferring style to 3D models is known, so instead of creating a new style transference algorithm from scratch, existing 2D transference methods have been analysed in order to decide on their possible adaptation for 3D style transfer. According to [Wilmot *et al.* , 2017], "Texture synthesis is the problem of statistically re-synthesising and input texture. Style transfer is similar: one statistically re-synthesises as input style exemplar S with the constraint that we also do not want the synthesised image O to deviate too much from a content image C". Even before the neural network breakthrough, there were some efforts in applying a texture from one source image (S) in a new image (O) with the content of another target image (C) [Efros & Freeman, 2001]. This solution required correspondence maps between the two reference images and the obtained results were not always visually appealing.

The main purpose of transferring a style would be to extract colour and texture from one image and generate a new one that integrates these extracted characteristics with the content of another image whilst delivering a visually appealing creative artefact. This could be described as taking features of style and combining them with the high level features of an image representing its semantic content, what could be achieved with a pixel-by-pixel analysis, producing the result of spatially-averaged colours. But the adaption of such approach would not however guarantee that the semantic meaning of the original content would be adequately preserved nor for that matter the core characteristics of the desired style itself. In fact, the intention would be to combine information from two different sources without having any knowledge on the specific percentages of information form each source or on how this combination could be achieved in the first place [Narayanan, 2017]. In the past, there was no artificial system that could possibly simulate the artist creation ability. Neural Networks provide the opportunity for analysing two independent images and generating a combination of them, as proven in the algorithm of Gatys et al. [Gatys *et al.* , 2015]. In this method, the style is efficiently transferred and its success is clearly demonstrated by the large number of citations the work has. All its limitations are clearly identified and with new suggesting improvements frequently released, because it is already a very well studied method.

The method proposed and developed in this dissertation for succeeding in 3D style transfer applies the algorithm developed by Gatys et al. [Gatys *et al.* , 2015] for the reasons described above. There are two main constraints to keep in mind to succeed in the style transference for 3D: (a) to apply any style to a model with the best result as possible and without requiring any extra control on the rendering of each result, the technique should be based on the work recently developed by

[6] The style should be smoothly applied to the 3D model. This problem will be fully identified in the following chapters.

[Gatys *et al.* , 2015] and (b) the result must be a visually pleasant 3D model with a chosen style applied, without requiring to change the model and without any kind of discontinuities.

Applying a 2D style transfer algorithm to a 3D model is a complex process because one has to guess how it will possibly behave in a 3D environment. In order to address this challenge, in this work, the chosen methodology was to first apply a style to the texture map of the 3D object and then reconstruct it into the 3D model, observing the obtained results and proposing solutions to avoid undesirable results, such as discontinuities in the frontiers of the texture islands. It is required to analyse in detail the taxonomy for 2D and 3D style transfer and choose a correct approach considering what has already been developed, and this process is described in chapter 2. In order to fully understand how the original algorithm of [Gatys *et al.* , 2015] works, chapter 3 has a detailed description with results obtained by testing different values of the variable parameters of the used code (open source code available in [Fzliu, 2015]), and a gathering of the critics to this algorithm is presented, with its possible specific influence on 3D transference. For more technical detail, a short explanation of Neural Networks is included in annex A. Also, the options of models for the neural networks are described in annex B and the best results acquired in the testing phase are presented in annex C. After understanding [Gatys *et al.* , 2015] 2D model, a first approach on how to directly implement the model for 3D style transfer has been performed yielding a set of difficulties that have been thoroughly mapped and possible solutions discussed in chapter 4. This Chapter presents the description on the practical implementation of 3D style transfer as well. Finally, the conclusions and future work are listed in chapter 5.

1.5 Thesis organisation

In this section it is presented a diagram describing how the format of this document follows the methodology of the work developed for this dissertation.

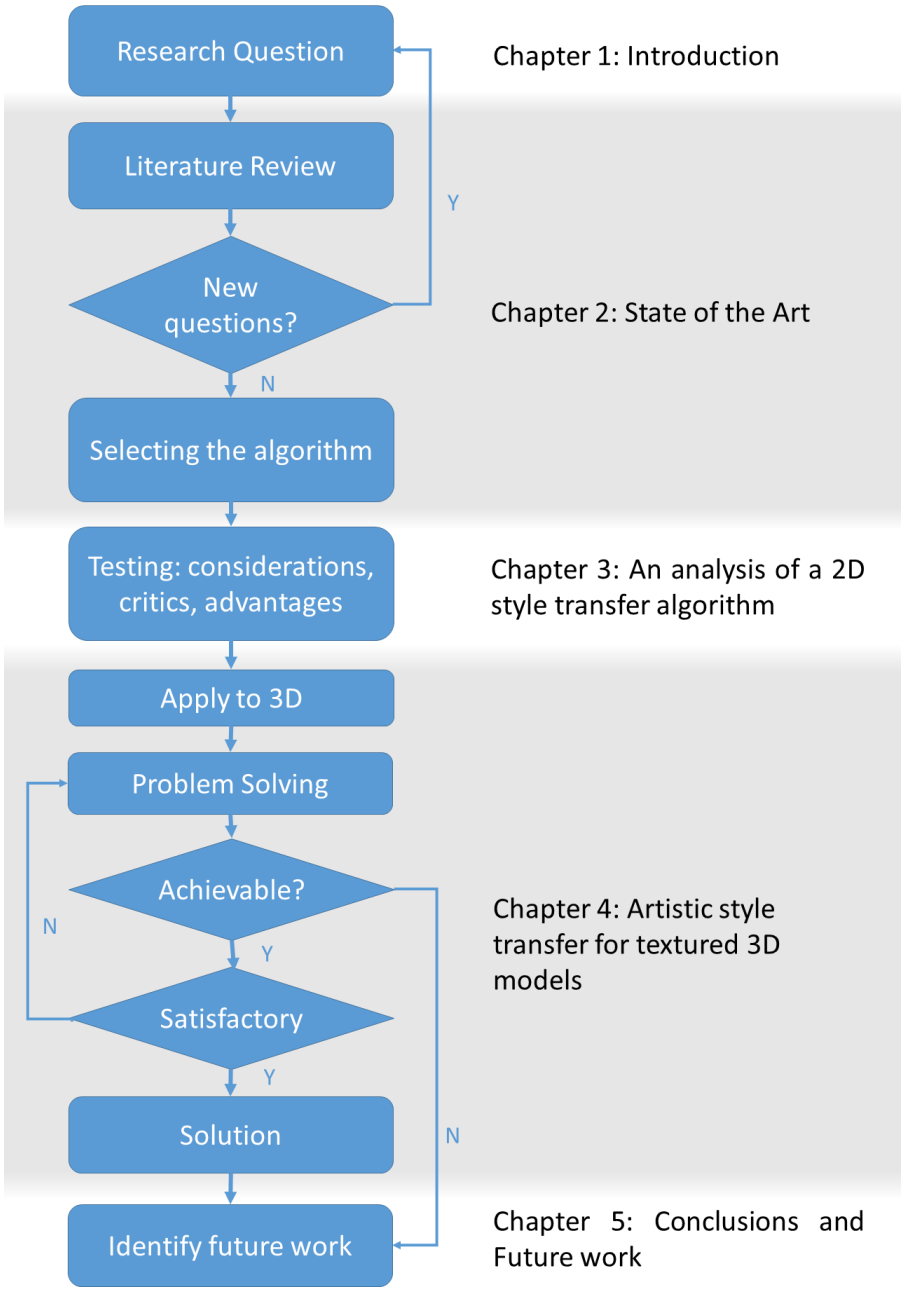

Figure 1.1: Diagram representation of this document

Chapter 2

State of the Art

2.1 Style Transfer as non-photorealistic rendering

The advent of photorealistic computer graphics in the early nineties fostered the development of alternative techniques for rendering in non-photorealistic styles [Kyprianidis *et al.* , 2013]. Nowadays, non-photorealistic rendering (NPR) has become a well-established field of knowledge and a vibrant area of research for visual communication, covering a number of expressive rendering styles, that may be organised into the following subsets [Kyprianidis *et al.* , 2013]: "exploded diagrams", false colour, and painting transfer techniques.

"Exploded diagrams" is a technique introduced by [Li *et al.* , 2008] for creating and viewing interactive 3D "exploded" views that allow users to explore the spatial relationships between specific parts of interest of complex 3D objects, such as mechanical assemblies, electronic devices, and built environments. In order to depict the internal structure of such objects, illustrators often create exploded views in which parts are separated (or "exploded") away from one another to reveal their components. However, traditional static exploded views have several limitations that can make it difficult for viewers to browse the parts of the objects and focus on different subsets of parts. Actually, most existing systems generate exploded views that are either meant to be viewed statically or provide limited interactive viewing controls. Several examples of these exist in our day to day life, ranging from assembly instructions to appliance brochures (e.g. Machines and Furniture). In contrast with these, [Li *et al.* , 2008] claim that their algorithm automatically determines the order and directions in which parts can explode without violating blocking constraints (i.e., without passing through each other) and further allows for high-level viewing tools for expanding and collapsing parts dynamically. In practice, the algorithm can possibly be used for creating interactive exploded views for a variety of models. The relevance of the algorithm presented by [Li *et al.* , 2008] for this thesis goes beyond the taxonomy attempted in this chapter for NPR approaches as the principles underlaying the technique developed by those authors are useful for better understanding the limitations of 3D style transfer, as explained in Chapter 4.

[Reinhard *et al.* , 2001] were certainly among the first to address the false colour problem, early this century. Put simply, the method proposed by these authors shifts and scales the pixel

values of the source image to match the mean and standard deviation from the target. Although this technique has proved to be successful for a large range of images, the quality of the results chiefly depends on the composition of the source and target images. In recent years, several colour transfer approaches have been suggested in the literature supported by a variety of techniques, namely, histogram matching [Neumann & Neumann, 2005], [Xiao & Ma, 2009], probability distribution function transference [Pitie *et al.* , 2005], just to mention a few. Despite the evolution perceived by the algorithms proposed after the work of [Reinhard *et al.* , 2001], a variety of limitations on colour transfer still deserve considerable discussion in the literature. An example of these is the extent of differences between the target image and the source image. For example, if the wonderful colours of Max Slevogt's "Unter den Linden" 1913 impressionist painting C.9 would be successfully transferred to an Alpine landscape, snow would inevitably appear green. Therefore, some type of external control (by the user) may have to be exerted on the algorithms, in order to prevent these problems to arise. The algorithm proposed by [Pouli & Reinhard, 2011] aims at overcoming pitfalls in colour transfer (or palette transfer, to put it according to Slevogt's painting) between two images. The authors suggest a novel progressive histogram reshaping approach while allowing the user to control the amount of matching [Pouli & Reinhard, 2011]. Obviously, the relevance of the attempts developed by the above researchers (and many others not mentioned in this paragraph) in order to successfully achieve colour transfer between two images are relevant both from the NPR classification aims of this section and from the objective of Chapter 4.

Painting transfer is an important subset of NPR nowadays, mostly due to the popularity gained from the ability to transfer styles of famous artistic paintings to current contents (e.g. images, photographs or videos). Accordingly, painting transfer is also mentioned as artistic rendering (AR) in the literature. Major developments in AR have focused on artistic stylising of 2D content, or 2D style transfer, also named in the literature as image-based artistic rendering (IB-AR). Early prototype 2D style transfer algorithms focused on stroke-based rendering paradigm with increasing levels of automation and sophistication in stroke placement. Those algorithms aimed at synthesising artistic renderings by incrementally composing virtual brush strokes whose colour, orientation, scale, and ordering were derived from semi [Haeberli, 1990] or fully automated processes [Litwinowicz, 1997], [Treavett & Chen, 1997]. But these techniques do not enable a choice of a style from a well known artist, as it is the objective stated in Chapter 4.

Another classification, but considering the most generic 2D style transfer techniques, can fall into three main categories [Selim *et al.* , 2016]: stroke based, texture transfer based and parts transfer based. Parts transfer based techniques consist on parsing the input image into different parts. A database of painted parts is then queried and used to reconstruct the final painting [Selim *et al.* , 2016].

Stroke based techniques have evolved from the early nineties semi-automated algorithms [Haeberli, 1990] to the current fully automated processes [Zeng *et al.* , 2009], [Hertzmann *et al.* , 2001], [Litwinowicz, 1997], [Lu *et al.* , 2010] successfully rendering the painting by simulating the brush-stroke placement process. The placement process is guided by a set of attributes such as its scale, orientation, colour and opacity.

Texture transfer based techniques [Hertzmann *et al.* , 2001], [Gatys *et al.* , 2015] modify the input image in a way to follow sample textures. Here ideas from textures synthesis are commonly used [Efros & Freeman, 2001].

Hertzmann et al. [Hertzmann *et al.* , 2001] presents a technique based on texture transfer that translates in processing images by example, a framework called "image analogies". Applying a filter to an image A, and so producing an image A', it is established a relationship in which A' represents A. So it is possible to find an "analogous" image B' that relates to B in the same way. Accordingly, this method produces appealing results in painting by example for style transfer.

The technique presented by Efros & Freeman [Efros & Freeman, 2001] is called image quilting, which consists on a fast and very simple texture synthesis algorithm, and so it is a good example of reproducing a texture. The quilting technique can also be applied for texture transfer, consisting on reproducing a texture from one image and applying it in another, but requires a correspondence map to do this. Spatial maps represent corresponding quantities over both the texture source image and a controlling target image, and for this reason it is not adaptable for any style and it is not automatic considering it is not user independent to map these correspondences.

All in all, Gatys et al. [Gatys *et al.* , 2015] introduced a rather recent technique using AI to identify style and separate this from content. The technique appears to be the most recent generic 2D style transfer technique in the literature and has as such deserved general acceptance from the research community. The approach followed by [Gatys *et al.* , 2015] for painting transfer uses CNNs [1].

2.2 Style Transfer using CNNs

Neural Networks are algorithms based on the animal neural system, designed to develop the capability of machine learning. These networks are basically a combination of small units, commonly referred as artificial neurons, that are organised by layers. These communicate, passing data from outputs to inputs of the following layers, in order to perform a specific function. [2]. In the case of the style transfer technique, these neurons read several images and compare their feature responses to generate an image with a combination of content and style, originated form different inputs. Using A.I. for this task has the great advantage of generating an output by a balanced combination of any image as content, and also any kind of style. It is a cognitive task, as the algorithm reads its inputs and reproduces a generated image, resulting of its previous knowledge.

In 2015, an article on style transfer using Neural Networks was published by Gatys et al. [Gatys *et al.* , 2015] and this had a great impact on Artistic Stylising State of the Art because the technique quickly went from a published research paper to an industry wide application, not only because it had outstanding results, but also due to a rapid evolution from minutes long run-time to instant rendering. It falls into the category of texture based technique for style transference and has impressive results considering the previous state of the art developments described before. Neural

[1] Convolutional Neural Networks. Consult appendix A for more detail
[2] Convolutional Neural Networks. Consult appendix A for more detail

Style transfer is a very recent technique used to produce a stylised image by the combination of the content of one input image and the style of another, fused together. When the first paper on the topic was published, the big question was what this could be used for and how this would change the Digital Arts and the Video Games industry. After this important mark, various papers were released using the same technique, with suggestions of improvements and alternatives for further applications.

Novak & Nikulin [Novak & Nikulin, 2016] give some suggestions on how to improve the [Gatys *et al.* , 2015] technique without changing the original structure too much, and the obtained results are better in simple examples of appliance, but still had some limitations, namely when there should be a separation between foreground and background, because an uniform background sometimes would get fragmented into differently stylised regions.

Also, Selim et al. [Selim *et al.* , 2016] proposed an algorithm to be applied exclusively to facial portraits. A user could select a face photograph and apply a style of a portrait painting, maintaining the artistic style but also the facial expressions of the person in the photograph. Selim et al. [Selim *et al.* , 2016] claims that the appliance of style developed by Gatys et al. [Gatys *et al.* , 2015] can alter the identity of the person in the content image. Nevertheless, Selim et al. [Selim *et al.* , 2016] avoids facial deformations adding a spatial constraint by means of gain maps.

Ikuta et al. [Ikuta *et al.* , 2016] presents an algorithm that learns a desired style of artwork from a collection of images and transfers this style to an arbitrary image. Ikuta et al. [Ikuta *et al.* , 2016] normalises features with the image size, what will describe the statistical amount that could be viewed as the frequency of a certain local feature among the entire image. Here it is considered that an artwork contains two types of visual features: (1) the features that are common in all of the images that share the same style and (2) features that are special in the certain instance of the artwork. And so, the style of an artwork is not characterised by the features of one work, but rather by the features that commonly appear within a collection of works. But this requires large dataset of images created in the same style, which can be a difficulty for a simple transference. Here is also highlighted a limitation in the first [Gatys *et al.* , 2015] algorithm, concerning the colour distribution, because the output image is restricted by the input texture image in colour, although it eliminated the need for region correspondences. This detail is said to cause unnatural results, because [Gatys *et al.* , 2015] does not distinguish the style of an artwork from the texture of a specific artwork.

A year later, a new article was released [Gatys *et al.* , 2016] with improvements by the same authors, for style transfer with colour, light and region constraints. The region constraints enables a control over what style can be applied to one specific region, for instance, guaranteeing that the sky of a an image has one specific texture, different from the rest of the style applied in the rest of the image. The colour constraints developed is either based on luminance-only transference or colour-histogram matching, avoiding unrealistic colours in the output image, considering the content that is being represented. Also, Gatys et al. [Gatys *et al.* , 2016] proposes a scale control for better output resolution.

Johnson et al. [Johnson *et al.* , 2016a] proposed a real-time style transfer solution, by training

feed-forward transformation networks for image transformation tasks, using perceptual loss functions that depend on high-level features from a pre-trained loss network. This method is based on the work of [Gatys *et al.* , 2015] and so the obtained results are similar, but are a lot faster to generate.

New improvements were suggested by Wilmot et al. [Wilmot *et al.* , 2017] now concerning limitations in texture quality, stability parameter tuning and lack of user controls in the algorithm of described in [Gatys *et al.* , 2015]. Wilmot et al. [Wilmot *et al.* , 2017] proposes a method that improves quality, convergence in fewer iterations and more stability over the optimisation process, using CNNs for style transfer and texture synthesis. In [Wilmot *et al.* , 2017] point of view, the problem definition for style transfer can be thought of as a broadening of the texture synthesis problem. Texture synthesis is the problem of statistically re-synthesising an input texture and style transfer is similar: one statistically re-synthesises an input style exemplar (S) with the constraint that we also do not want the synthesised image (O) to deviate too much from a content image (C). For guaranteeing stability over the optimisation process, [Wilmot *et al.* , 2017] proposes an histogram matching technique: to remap the synthesised output activations to match the activations of the input source texture. Wimot et al. [Wilmot *et al.* , 2017] proposes a new formula for the loss function based on histogram matching. Firstly, transforming the synthesised layer-wise feature activations so that their histograms match the corresponding histograms of the input source texture. This matching must be performed once for each histogram loss encountered during back-propagation. Secondly, adding a loss between the original activations and activations after histogram matching. And so the loss function is a sum of the style loss, the histogram loss, the content loss and the total variation loss from [Johnson et al.] for improving smoothness. Wilmot et al. [Wilmot *et al.* , 2017] uses image pyramids [3] for better resolution, improving the image quality of the output. Because one image may have more than one completely different textures laid out in a scene, [Wilmot *et al.* , 2017] suggests painting by numbers. The control of the parameters that can be variable in the code is automatic, having 4 weights of each type of loss. Although there are presented very interesting results, [Wilmot *et al.* , 2017] guarantees the spatial constraints with manually painted masks.

It is evident that this way of applying style using Convolutional Neural Networks became popular in the last few years, but mostly for transference between 2D images. The research published by [Ruder *et al.* , 2016b] presents a method to transfer style to video, with new initialisations of the system and new loss functions. Based on the same loss minimisation presented by [Gatys *et al.* , 2015], temporal constraints are added to guarantee stronger short and long-term temporal consistencies, enabling the production of stable stylised videos. So this research showed a first step into applying this style transfer techniques to other systems that are not solely based on 2D images. Unfortunately, considering specifically 3D style transfer, only a few research have been released on this topic.

[3] An image pyramid is a collection of images, all arising from a single original image, that are successively downsampled until some desired stopping point is reached. In this case, the Laplacian pyramid is used to reconstruct an upsampled image from an image lower in the pyramid (with less resolution) [Burt & Adelson, 1983]

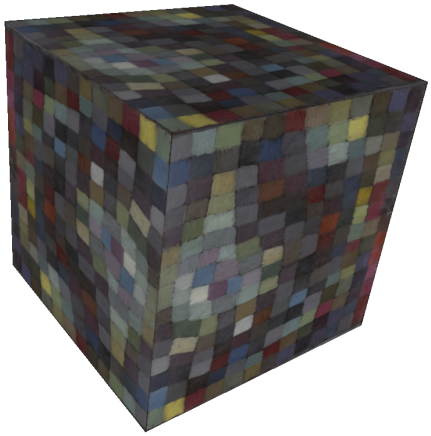

Figure 2.1: 3D model of a cube, with image from C.7

```
v -0.5 -0.5  0.5
v  0.5 -0.5  0.5
v -0.5  0.5  0.5
v  0.5  0.5  0.5
v -0.5  0.5 -0.5
v  0.5  0.5 -0.5
v -0.5 -0.5 -0.5
v  0.5 -0.5 -0.5

vt  0  0
vt  1  0
vt  0  1
vt  1  1

vn  0  0  1
vn  0  1  0
vn  0  0 -1
vn  0 -1  0
vn  1  0  0
vn -1  0  0

s 1
f 1/1/1 2/2/1 3/3/1
f 3/3/1 2/2/1 4/4/1
s 2
f 3/1/2 4/2/2 5/3/2
f 5/3/2 4/2/2 6/4/2
s 3
f 5/4/3 6/3/3 7/2/3
f 7/2/3 6/3/3 8/1/3
s 4
f 7/1/4 8/2/4 1/3/4
f 1/3/4 8/2/4 2/4/4
s 5
f 2/1/5 8/2/5 4/3/5
f 4/3/5 8/2/5 6/4/5
s 6
f 7/1/6 1/2/6 5/3/6
f 5/3/6 1/2/6 3/4/6
```

Figure 2.2: Object file of 2.1

2.3 A basic view of 3D modelling

Focusing on 3D style transfer, the main aspects on how 3D modelling works should be highlighted. The first thing one should know about 3D modelling is that a 3D model can be built by defining a certain number of vertices and then connecting them creating polygonal faces. The more complex the models are, the more vertices will have to be defined. Files containing the model description information will be further named object files [4].

A basic example of a model would be a representation of a cube, as depicted in figure 2.1. To generate this model, the list of vertices used is shown in figure 2.2, where the first list with a 'v' index indicates the vertices of the cube, the 'vt' list concerns the texture coordinates and the 'vn' list represents the vertex normals.

The vertices of the cube are described with the (x,y,z) coordinates in a 3D space and the order of appearance of these vertices will influence the rest of the description model, as it will be explained later on.

The texture coordinates are used as a map for correspondence between a two-dimensional image and the three-dimensional vertices model, as shown in figure 2.3. As such, the values established in the code are related to an (u,v) axis concerning the real distances in the texture two-dimensional image. This means the texture coordinates will be composed by a list of (x,y) coordinates, which values will be always contained in the range [0,1], and this is represented in figure 2.4.

The vertex normals are the light reference. In a simplified way, in a certain vertex, the angle between the incident light vector and the normal to the surface vector will indicate the inclination of that surface towards the light source and so, it assigns the illumination value in this point, as

[4] Often called .obj files

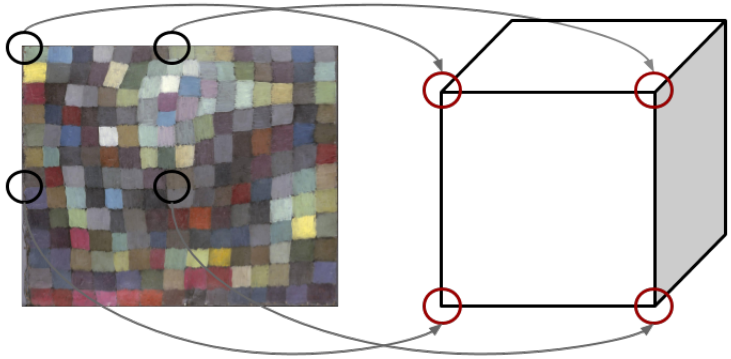

Figure 2.3: Texture correspondences representation, with image from C.7

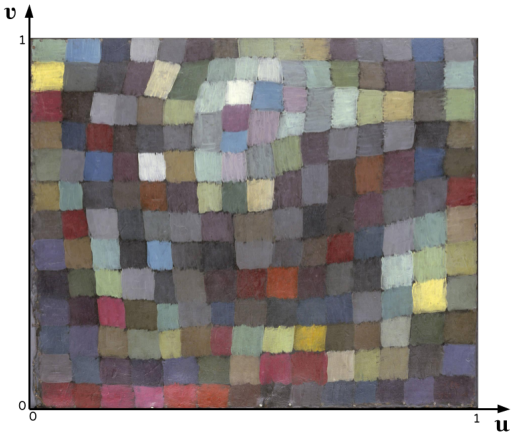

Figure 2.4: Texture mapping representation, with image from C.7

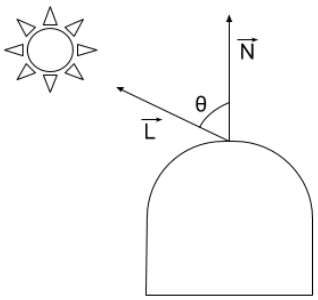

Figure 2.5: Light exposure representation

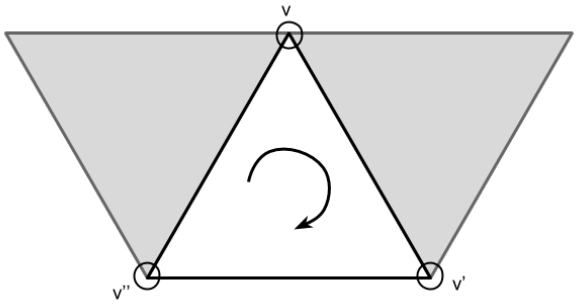

Figure 2.6: Texture vertices' order

shown in figure 2.5. So by defining the vectors describing the surface orientation of the surface (also known as normals), the model is fully described concerning the light exposure.

Finally, the list of polygonal faces is presented with an 'f' in the object file, for establishing the connections between the three kinds of vertices, forming the surfaces. These are divided in smoothing groups identified by the letter 's', which correspond to a smooth mesh of polygons and this allows the 3D modelling software to estimate the surface normal at any point on the mesh by averaging the vertex normals. The polygonal faces indicate the correspondence between 3D vertices, texture and normals, by order of appearance in the code, and combining these, they form triangles, represented in figure 2.6. The combination of these triangles, or surfaces, builds a 3D model fully described by position, light and texture. And so, with this code, the main definition of a 3D model is complete, and this forms an object file, used by a 3D modelling software, that will render the final shape.

2.4 Style Transfer to 3D models

A few researches can be found on 3D style transfer, as [Ma *et al.* , 2014], that refers a way of generating a new image with a selected style. The process requires 3 reference images: one for source, another as the target and the exemplar image. The source image is an example of style and the target represents an image with a different content, but with the same style as the source image. The exemplar image is the required style, so the output will have the same structure as the target image, but with the style of the exemplar. But this system could not be applied for the purpose described in Chapter 4 because it would require that to choose a style and a content image, the original image of the initial style that produced it would be needed for the system to be complete. So this works as a comparison between the changes of an original image and its style, and so in this system there's a lack of the source image. Also, the three images have to be similar in terms of structure, and this does not apply to the terms of liberty of choice of a style image. Lastly, the 3 images have to be equally aligned and with the same orientation for this system to work, which reveals as an impossibility for the system described in Chapter 4, putting the liberty of choice for style at stake.

Some other works related to style transfer applied to interior design [Chen *et al.* , 2016], [Nguyen *et al.* , 2012] and, more specifically, furniture and decoration objects [Han *et al.* , 2015] were proposed, but mostly dedicated to colour transference and shape modification.

Another research was released in [Fišer *et al.* , 2016], a method based on light propagation in the 3D scene. They consider that a limiting factor of previous techniques was that those relied mainly on colour information to determine the stylised appearance, what leaves them unable to distinguish among different regions that have similar colours. [Fišer *et al.* , 2016] states that there should be as much attention to lighting effects as they to colour when painting a 3D scene. This approach allows illumination-dependent stylising, computing light propagation in a simple scene, letting artist provide a corresponding painting in arbitrary styles. So they use a comparison between a simple model and then transfer the style applied by the artist to a more complex model

but of similar light exposure. The transference of style is made by finding appropriate regions in the exemplar painting to transfer appearance from. The shadow, highlight and diffuse regions in the synthesis result exhibit similar visual style to the corresponding regions in the exemplar. But for the system described in Chapter 4, one of the main targets is the freedom of choice of a style by the user, so that there is no dependence on artistic capabilities. As the [Fišer *et al.* , 2016] assumes an artist drawing a simple 3D model and the resulting style is applied to a more complex model using the illumination information to guide the relationships.

In [Lu *et al.* , 2010] it is presented an approach adapted not only for 2D style transfer but also for video and 3D models. Although the research point out interesting developments, it is based on stroke transfer techniques, and so does not meet the goals described in Chapter 4.

2.5 Conclusion

Considering the purpose of transferring a style from an art work to a 3D model, fulfilling the objectives set for this dissertation as presented in Chapter 4, it can be concluded that most of the 3D appliances highlighted in the previous section do not meet these goals, as they are based in colour and shape alteration, and not in artistic styles transference [Ma *et al.* , 2014], [Chen *et al.* , 2016], [Nguyen *et al.* , 2012], [Han *et al.* , 2015]. Nevertheless, [Fišer *et al.* , 2016] purposes an interesting solution for applying an artistic style to 3D models, but requires the work of an artist dedicated to an exemplar model, for further appliance in more complex forms. One of the objectives of this dissertation is not requiring an artist to create stylised 3D models, so the selected styles can not depend on any exemplar models, as they can not be tailor-made.

[Lu *et al.* , 2010] presents a strategy that can be applied to 3D models, producing significant results. But comparing the outputs in terms of style with the ones presented by [Gatys *et al.* , 2015], they are not as visually appealing. And so, from the analysed research described above, it is not acknowledged a system that can apply style from any artwork to a 3D model successfully, producing results as good as the ones that are being presented in several researches for 2D images.

The method presented in this dissertation seeks to overcome this problem as it is stated in [Lu *et al.* , 2010] when applied to 3D models, where it is considered the information given by the coordinates of the models as an indicator for the style transference, with the method of [Gatys *et al.* , 2015], using Convolutional Neural Networks to read content and style independently and generating a new image as a combination of the two components. The purpose of this is to produce 3D stylised models qualitatively comparable to the results produced in the 2D field.

Chapter 3

An analysis of a 2D style transfer algorithm

After analysing some alternative solutions for transferring styles from one image to another, the chosen method for this project was the one developed by [Gatys *et al.* , 2015] as it is a very recent algorithm referenced numerous times for various applications [Luan *et al.* , 2017], [Ikuta *et al.* , 2016], [Gatys *et al.* , 2016], [Johnson *et al.* , 2016b], [Ruder *et al.* , 2016a], [Huang & Belongie, 2017].

This algorithm is able to generate a new image where it applies the content and the style from two different sources. As it uses a Convolutional Neural Network trained on object recognition, it is able to identify and maintain the main aspects of the target content. The relevance of using Neural Networks for this task of style transference is that it can be applied to different situations without requiring new development for adaptation, i.e. the user may select different images of content and style and this algorithm will generate a new image without requiring any changes in the code.

As stated in the previous chapters, for this dissertation, the style transfer algorithm developed by Gatys et al. [Gatys *et al.* , 2016] will be adapted to apply style from a work of art to a texture map, and so it is relevant to analyse the original code and interpret the obtained results by changing some variables of the system.

3.1 Style

So far, it was used the expression “style transfer”, but there is some discussion in the state of the art on this matter, considering the definition of style. The style of a work of art can be defined differently and sometimes it may depend on the area of application considered. In computer science, creating digital paintings requires the interpretation of style, either as statistics of colour patterns [Gatys *et al.* , 2016], or patches and illumination information [Fišer *et al.* , 2016], or features [Ikuta *et al.* , 2016], or artistic rendering primitives: regions, strokes, stipples and tiles [Kyprianidis *et al.* , 2013]. According to Ernst Gombrich, an art historian from the twentieth century, in fine arts,

Figure 3.1: Mandrill

Figure 3.2: Mandrill with Les Demoiselles D'Avignon's style C.1

style comprises even more detail in its definition: "Style is any distinctive, and therefore recognisable, way in which an act is performed or an artefact made or ought to be performed and made" . As such, style could be considered as a combination of more than texture, colour, shading or light, but also geometry, composition, perspective, measurement, scaling and volumetry. In addition, there are some aspects concerning the content that depending on the history epoch of the painting, would differ significantly: what is represented, how people dressed, what is the focus of the painting and what level of dramatisation is showed in the characters. Unfortunately, it is impossible to guarantee all these characteristics when generating a digital painting using the Style Transfer method[1]. Despite all of the details of a work of art, the transference of style to another image can only concern what the Neural Networks can read in its features, without changing the original content. Taking as an example of this Style Transfer method concerning what is considered as style, it is shown in Figure 3.2 the output of the combination of the Mandrill image as content 3.1 and the painting Les Demoiselles D'Avignon, by Pablo Picasso, C.1 as the input style.

The painting referenced above, Les Demoiselles D'Avignon C.1, is an iconic work of art of the Cubist Movement, born in the twentieth century. More specifically, this painting is considered by various authors the first work of art marking the beginning of Cubism. The movement had two periods, the analytic and the synthetic, but the selected paintings on this document focus on the first one. Concerning painting, this is characterised by figures constructed with geometric shapes, showing multiples points of view simultaneously in the same painting, flattered figures with minimal three-dimensionality, sharp figures as opposed to rounded volumes, absence of perspective, with muted and dull colours. The main reason for combining such figures was to put vision and memory working together, in a way to express total visual understanding, by fragmentation of the figure and rebuilding it to provoke the perception of complete information, although it is still a two-dimensional image. One can see these characteristic features in Les Demoiselles D'Avignon, but applying its style to another image, not all the traits are kept. This would be expected in view

[1]Because that would imply changing the content geometrically or even what is being represented

Figure 3.3: The Portuguese Braques [1911]

Figure 3.4: Mandrill with The Portuguese's style

of the Style Transfer algorithm works [Gatys *et al.* , 2016] [2]. Nevertheless, some of the main features are coherent with the original style image and it is still distinguishable the Cubism characteristics in the presented result 3.2. For instance, the sharp geometric figures and the colours are maintained, although the perspective and the three-dimensionality are loyal to the content image, and especially the sense of multiple points of view is lost. Considering the information given to the Neural Network, this is not perhaps so surprising because there is no data about different points of view. Another example of this is presented in Figure 3.4 where another iconic Cubist painting was selected for the style source, applied to the same content. The results are similar, proving the coherence of perspective and three-dimensionality with the content image, and not the style, but changing the colours and type of geometric figures, in respect to the style chosen. Also, it is important to point out that there is no deformation in the original figure, due to a feature recognition of the content form, and only applying the traces of the style painting.

However, this methodology to generate a digital artistic images can be applied to any style image of choice, and although the results may not correspond to every technique aspect of an artistic style, it produces a visually appealing result, with the main characteristics of a style to reach a new representation of a certain content. And so, by using the Style Transfer algorithm [Gatys *et al.* , 2016], we may not be able to exactly predict the output but, considering a certain style, it is possible to have an idea of what will be passed on as style. An illustrative example of this is in Figure 3.6, where the style image is Girl with a Pearl Earring, by J. Vermeer C.2, and a photograph of Scarlett Johansson interpreting the role of the person represented in this same painting, in Girl with a Pearl Earring (2003). The choice of this example has the goal of comparing the output of the algorithm in use with what would be expected from the original painting, in a sort of inverted logic.

Regarding the results obtained in Figure 3.6, the transferred style is based on statistics of colours, and so the lighter and darker values are evident in the output, the colours are adapted

[2] see Appendix A

Figure 3.5: Scarlett Johansson in Girl with a Pearl Earring Buitendijk [2003]

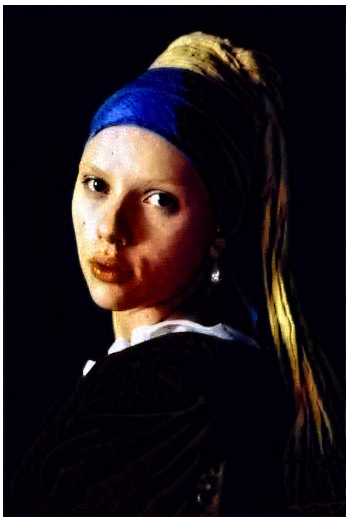

Figure 3.6: Scarlett Johansson's photograph with Girl with a Pearl Earring's style

evenly and the traces of the artist are also reflected on the output. All in all, although this is not a method to generate images as if they were painted by the original artists, it produces results that meet the artistic techniques significantly, creating images visually appealing and that can be used not only for social media, as it is today by means of mobile applications Moiseenkov [2016], but also can be a good starting point to create new pieces, as it is a good tool for digital artists.

3.2 The 2D Style Transfer algorithm

Convolutional Neural Networks are mostly used for image classification, which is the task of taking an image as an input and outputting a class or a probability of classes that best describes the image. But as CNNs are nowadays used for a diversity of tasks, the Transfer Learning technique has been commonly applied, which consists on a simplification to avoid the need of large datasets and the computational time to train a network from scratch [Karpathy, 2017c]. Instead, Transfer Learning involves using a trained model on a different dataset and adapting it to the considered problem. In the case of the Style Transfer algorithm developed by Gatys et al. [Gatys *et al.* , 2015], they chose models trained on object recognition, such as VGG-16 and VGG-19 [3], and adapted the algorithm to the information captured by the network, which in this case were the input images feature responses.

The main goal of this system was to separate the content of an image from its style, in order to successfully transfer the textures and colours of one painting to an output which content comes from a different image, that enters the system as an input as well. The choice of a Neural Network for the task is because this project belongs to the field of visual perception, and Convolutional Neural Networks are the most powerful visual models in image processing tasks.

[3] See Appendix B

As it is explained in Appendix A, CNNs consist of layers of small computational units that process visual information hierarchically in a feed forward way. Each layer of units can be understood as a collection of image filters, each of which extracts a certain feature from the image. The outputs from each layer will be differently filtered versions of the input image, and these are called feature maps. Whereas the low-level filters concentrate in finding small details, the feature responses in the higher layers capture high-level content[4], in terms of objects and their arrangement in the input, but do not constrain the exact pixel values. This style transfer algorithm focuses on creating a new image that simultaneously matches the content representation of the input photograph and the style representation (colours and local structures) of the respective input work of art, by minimising an objective function. Effectively, this renders the photograph in a different style, such that the appearance of the generated image resembles the work of art, even though it shows the same content as the photograph.

The network itself is built in the Caffe framework [Jia *et al.* , 2014], where the operations for each layer are defined and where blobs are created, which are the most basic unities that enable storing data and communicating. On a second stage, the network has to be fully described by number and types of layers, so it may follow different models, and this is what specifies which layers are used for evolving in the recognition of content, starting on the small details up to the arrangement of the objects in the picture. In this document, VGG-16 and VGG-19 are emphasised in Appendix B.

After choosing two images representing content and style, and the model of the Neural Network, the third stage, which is the style transfer system, will process the information from some chosen layers (with different weights for each layer) and produce a new image resulting of the combination of the two inputs. It is important to highlight that this system uses the weights of the network model that were previously calculated for object recognition tasks. This is an alternative to using techniques that would directly manipulate the pixel representation on an image. The advantage of using an object recognition approach is that the information is represented on the high-level content of the image.

In short, the content of the generated image is calculated by comparing the feature responses (i.e. the feature maps values) between the content input image and the initialisation image, which will be transformed in every iteration to be approximated to the content and style of the other images. This method is represented in Figure 3.7. The style of the generated image is obtained by comparing correlations between feature maps of each layer, and this is illustrated in Figure 3.8. The style is a multi-scale representation that includes multiple layers of the network. The choice of the layers to be considered and of the number of layers to be correlated produces different visual experiences. In contrast to the content recognition, the analysis is based on a relation between the response to diverse features. This means the style is stationary: it is independent of specific features and their locations on the picture, what matters are the relations between those features.

[4]Feature responses in high level layers can be referred as content representation, but this term is referred to the CNN in a general basis

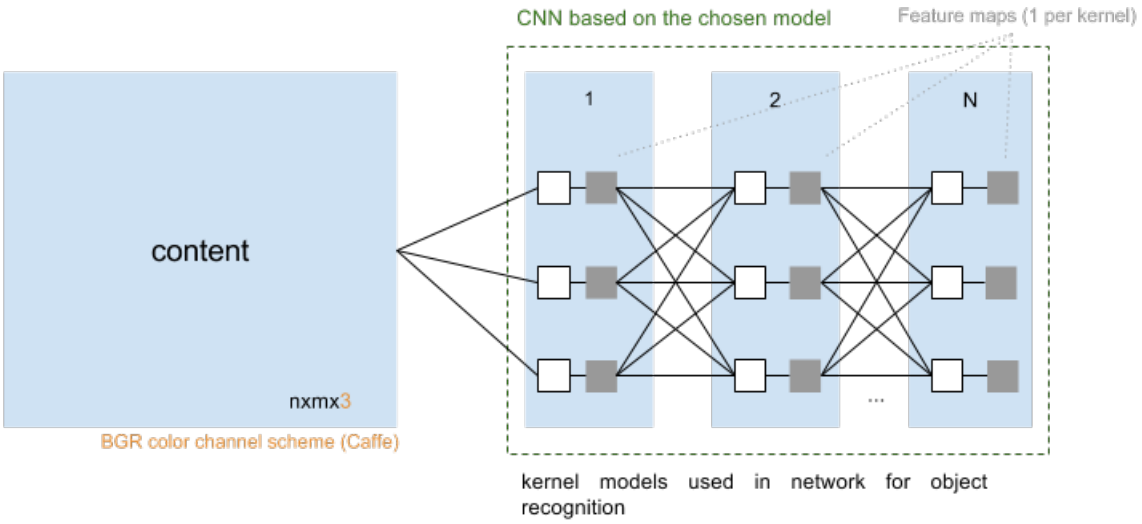

Figure 3.7: Content approximation

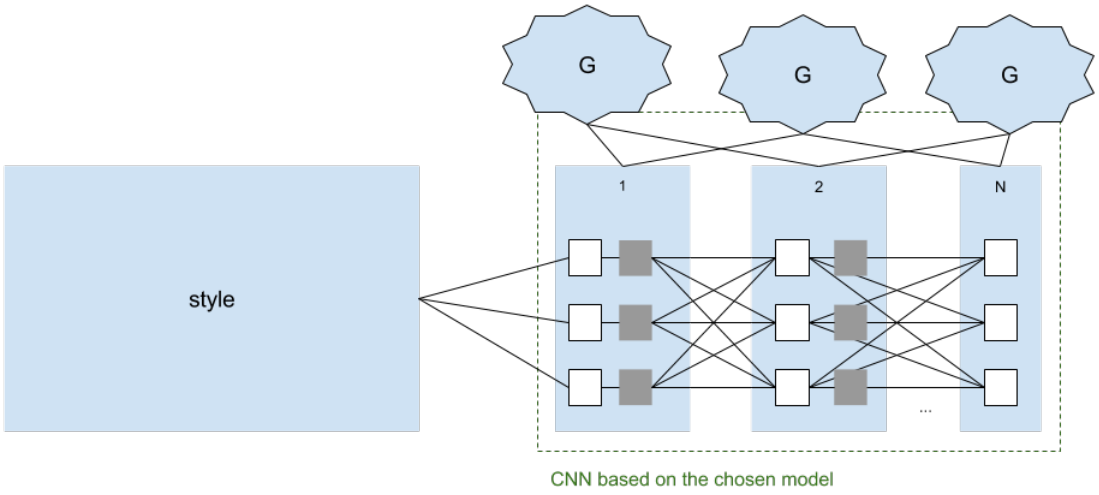

Figure 3.8: Style approximation

Therefore, the content and the style are both product of feature responses of a same network. The approximation of the generated image is achieved by minimising a loss function, presented in Equation 3.1, in order to have a balance of the relationships between the generated image and the style or content. The ratios α and β represent the influence of each portion, to control the emphasis of either the content or the style, I is the generated texture map, T is the original texture map and S is the style image.

$$\mathscr{L}(I,T,S)_{total} = \alpha \cdot \mathscr{L}(I,T)_{content} + \beta \cdot \mathscr{L}(I,S)_{style} \tag{3.1}$$

$$\mathscr{L}(I,T)_{content} = \sum_{l=0}^{L} \frac{1}{2M_l N_l} \sum_{i}^{N} \sum_{j}^{M} \left(F_{ij}^{l} - P_{ij}^{l}\right)^2 \tag{3.2}$$

$$\mathscr{L}(I,S)_{style} = \sum_{l=0}^{L} w_l \cdot E_l \tag{3.3}$$

$$E_l = \frac{1}{4M_l^2 N_l^2} \sum_{i}^{N} \sum_{j}^{M} \left(G_{ij}^{l} - A_{ij}^{l}\right)^2 \tag{3.4}$$

$$G_{ij}^{l} = \sum_{k} \left(F_{ik}^{l} \cdot F_{jk}^{l}\right) \tag{3.5}$$

More specifically, the Equations 3.2 and 3.3 are the detailed representation of the content loss function $\mathscr{L}_{content}$ and style $\mathscr{L}_{style}$. The minimisation of the style loss represents a reduction on the distance between the generated image and the content. This process consists on a minimisation of the loss function with a gradient approximation, given by the backpropagation step, so that the values in the generated image can be changed in the right direction and amount (increasing or decreasing pixel values). The content loss is represented in Equation 3.2, where M_l is the dimension of the feature map (i.e. $width \times height$), and N_l is the number of distinct filters, that coincides with the number o feature maps, in one layer. These are used to normalise the value of the loss function so that it does not depend on the characteristics of each layer. Both of these values are only constant among a same layer. F_{ij} is the activation of the i^{th} filter at position $j \in \mathbb{R}^2$ in the layer $l \in \mathbb{N}$, and P_{ij} is the feature representation of the original image (i.e. the activations of the content image). The gradient of the content loss in respect to the generated image is shown in Equation 3.6.

As for the style, the approximation is slightly difference, since its representation is based on correlations between feature representations. The style loss function is presented in Equation 3.3, where the contribution of each layer E_l for the total loss is multiplied by a variable weight w_l. This contribution is represented in Equation 3.4, where M_l and N_l have the same meaning as before, G_{ij}^{l} is the correlation between feature representations of the generated image, as shown in Equation 3.5, and A_{ij}^{l} has the same principle applied for the style image. And so, the reduction of each contribution E_l is processed by minimising the total loss function, and the direction of the changeable values of the generated image is given by the gradient of E_l in respect to the generated image, shown in Equation 3.7. Both equations 3.6 and 3.7 indicate the amount and direction of the changing values for the generated image, and these must me multiplied by the the weights α and β as well to guarantee the evidence of both constraints in the minimisation process.

$$\frac{\partial \mathscr{L}_{content}}{\partial F_{ij}^{l}} = \begin{cases} (F^l - P^l)_{ij}\,, & if\ F_{ij}^{l} > 0 \\ 0\,, & if\ F_{ij}^{l} < 0 \end{cases} \tag{3.6}$$

$$\frac{\partial E_l}{\partial F_{ij}^{l}} = \begin{cases} \frac{1}{N_l^2 M_l^2}((F^l)^T (G^l - A^l))_{ij}\,, & if\ F_{ij}^{l} > 0 \\ 0\,, & if\ F_{ij}^{l} < 0 \end{cases} \tag{3.7}$$

Therefore, to generate the output image, the distance between the generated image and the content and style representations is minimised by the process described above. This means that an initialisation image is changed according to its feature responses, so that the results in the layers are close to the ones obtained for content and their correlations are similar to the style correlations obtained before. This initialisation image will be the generated image in each iteration, changed accordingly to do an approximation that will never be completely equal to content feature responses and style correlation of responses, but a balance between them. The minimisation of the loss function, with both content and style comparison, is a process that is calculated in every iteration until an optimal solution is met or until a predetermined number of iterations is complete.

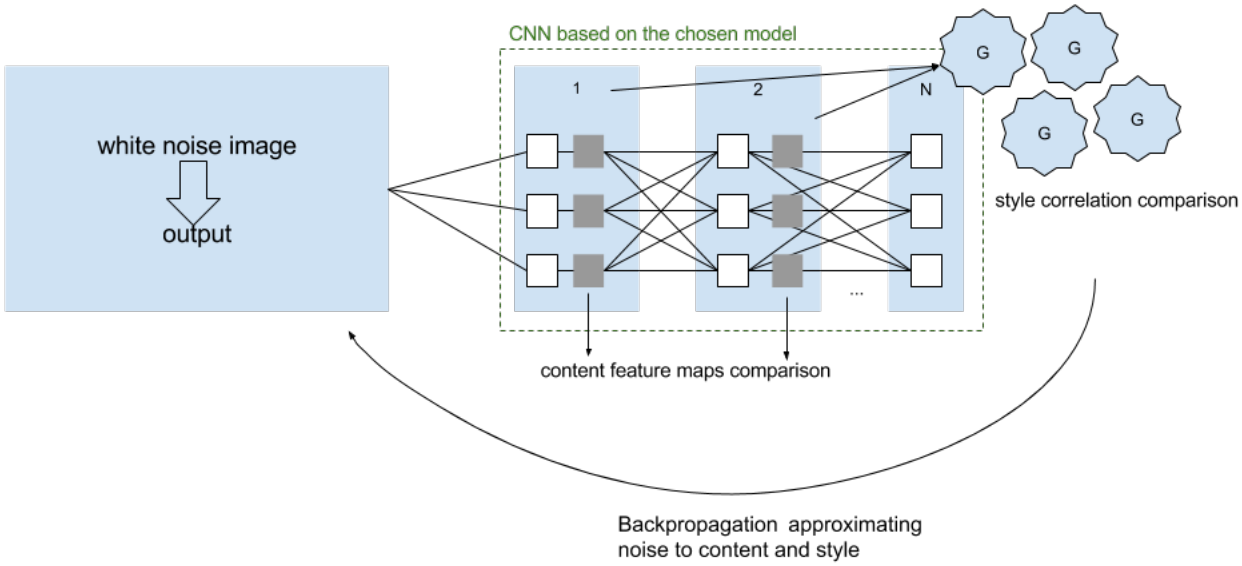

Figure 3.9: System illustration

This number of iterations is one of the controlling variables of the algorithm developed by Gatys et al. [Gatys *et al.* , 2015]. Actually, the system combines various parameters, and depending on the values selected for these, it leads to different results. Since this dissertation focuses on the development of a system based on the solution proposed by Gatys et al. [Gatys *et al.* , 2015], understanding how these variables influence the output and how to tune them properly to meet the desirable goals is of utmost importance. Regarding this, the results of the conducted tests presented in this document used the model VGG-16[5], with content and style taken from a combination between feature responses from weighted convolutional layers, as shown in Table (3.1). VGG-16 is a model that is composed by 5 convolutional layers and 5 pooling layers. The first convolutional layer has 2 repetitions ($conv_{1\_1}$ and $conv_{1\_2}$), as does the second layer ($conv_{2\_1}$ and $conv_{2\_2}$), whereas the three following layers have three repetitions ($conv_{3\_1}$, $conv_{3\_2}$, $conv_{3\_3}$, $conv_{4\_1}$, $conv_{4\_2}$, $conv_{4\_3}$, $conv_{5\_1}$, $conv_{5\_2}$, $conv_{5\_3}$). In Table 3.1 the third repetitions are not

[5]See Appendix A and B

presented because they all have a weight of zero at this point. These convolutional layers are separated by pooling layers, responsible for the downsampling, to avoid large amounts of information in the feature maps. This algorithm uses an average pooling operation and does not use any of the the fully-connected layers.

Table 3.1: Distribution of weights in the selected convolutional layers

Convolutional Layers	1_1	1_2	2_1	2_2	3_1	3_2	4_1	4_2	5_1	5_2
Content								100%		
Style	20%		20%		20%		20%		20%	

By varying the weighted convolutional layers, different results can be obtained. In Figure 3.10 it is presented to comparison between the results obtained by changing the selected layers for style identification, more specifically by using the layer $conv_{4\_2}$ for content representation[6] and a single convolutional layer for the style correlations. For these results, the selected style image is Figure C.3 and the content image is Figure C.4.

The images presented in Figure 3.10 were generated with 50 iterations. It is possible to observe that with lower levels of convolutional layers, the results are concentrated in colour and spot information, and not as much related to the tracing, so this reproduces images that resemble the pointillism technique, and not the work of art's style chosen. Using higher layers, the patterns obtained have a strong evidence of strokes, but disregard the colours and forms of the style, producing images with strong evidence of content, but small alterations concerning the style. And so, in general, this algorithm would not get the same results if it depended only on one single layer for the style. As such, it is by a combination of layers that the tracings and colours from this one work of art are obtained. Figure 3.11 presents the results of some possible combinations of only lower, middle or higher levels, by sharing weights of 50%.

The best results obtained are due to a combination of correlations of the five different convolutional layers, as shown in Figure 3.12, either with the first repetitions or the second repetitions, using the weights presented in Table 3.1.

The results obtained for the comparison between the repetitions are very similar, and as long as choosing a combination of the 5 layers, concerning the visual appearance of the output, the difference between the choice of different repetitions is not relevant for the goals of this dissertation.

Considering the choice of the reference content layer, it is important that it has the presence of the main information on the input image, but also that a good influence of style is permitted. A good intermediate for this would be the fourth convolutional layer, as shown in Figure 3.13, using as content the image in Figure C.5 and style in Figure C.6. As for the number of the repetition, as stated before, the difference is not relevant when the number of iterations is adequate.

So, considering the selection of the convolutional layers for either style correlation or content approximation, all the images presented in this dissertation obtained by using this Deep Style Transfer algorithm developed by Gatys et al. [Gatys *et al.* , 2015] use the layers presented in Table 3.1 because after testing some alternative combinations, the conclusion was that this would

[6]This means the comparison between the feature responses is done only in this layer concerning the content

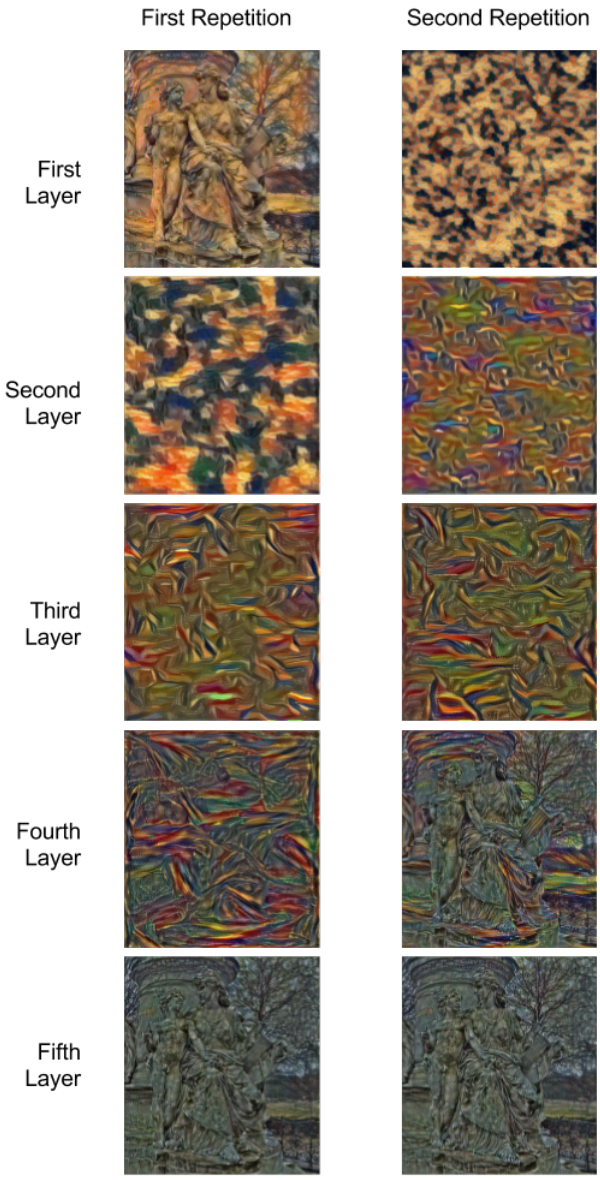

Figure 3.10: Style Representation using different convolutional layers and repetitions

be the best selection for the goals established in this project. But the system has other variables designed to control the output desired, and this is represented schematically in Figure 3.14, where the considered values of these parameters are presented as an example of a possible solution: the number of iterations equal to 512, the style ratio at 1×10^4 and a chosen initialisation image.

Firstly, the number of iterations is the number of approximations, so in every iteration the total loss function is slightly minimised. And so, the number of iterations can be responsible for enabling a better approximation to a good balance of style and content, as this minimisation will continue until an optimal solution is achieved or until the number of iterations is completed. This parameter influences the results notoriously and the comparison between the results obtained by

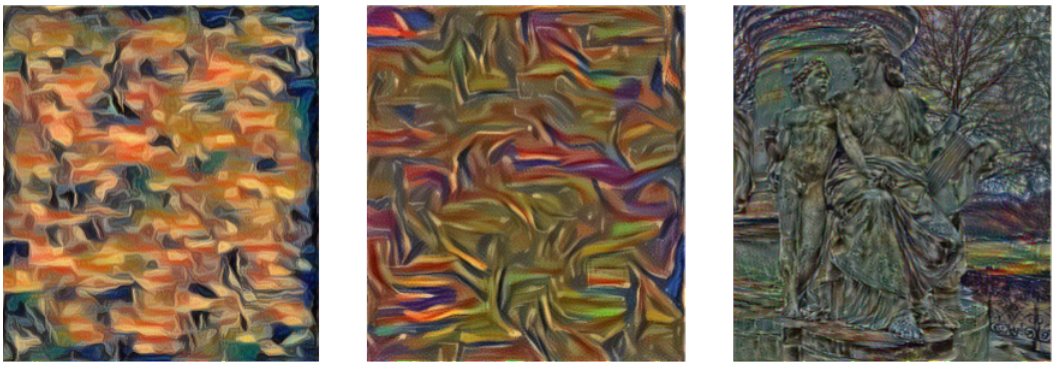

Figure 3.11: Style Representation using more than one convolutional layer and repetitions

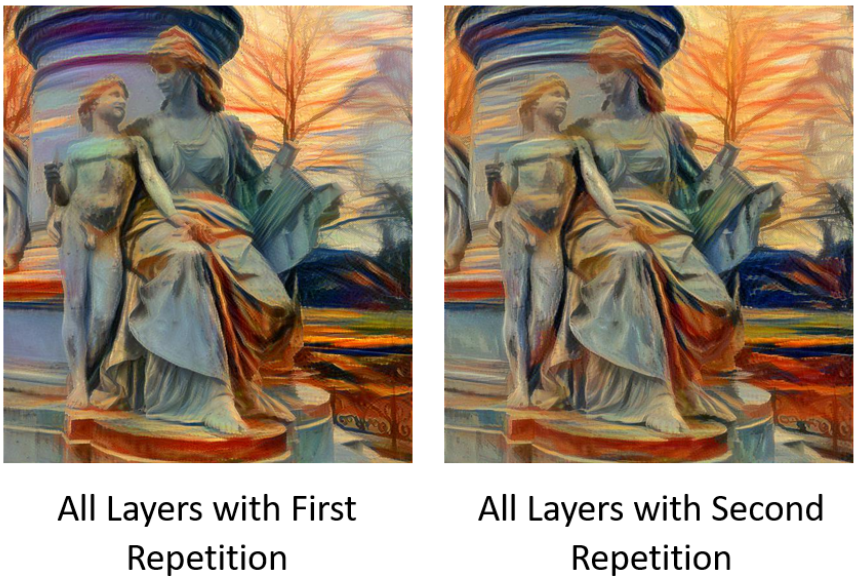

Figure 3.12: Style Representation using all convolutional layers and different repetitions

varying the number of iterations can be observed in the images presented in Figure 3.15, where the highest number of iterations[7] led to a more visually appealing result, using as content image Figure C.13 and style Figure C.12.

Secondly, the style ratio represents the emphasis of the textures on matching the content and the style for the output image representing a rate between the weight α and β presented before:

[7]The number 2^9 is a standard value for a good estimation of the approximation, as suggested in [Fzliu, 2015]

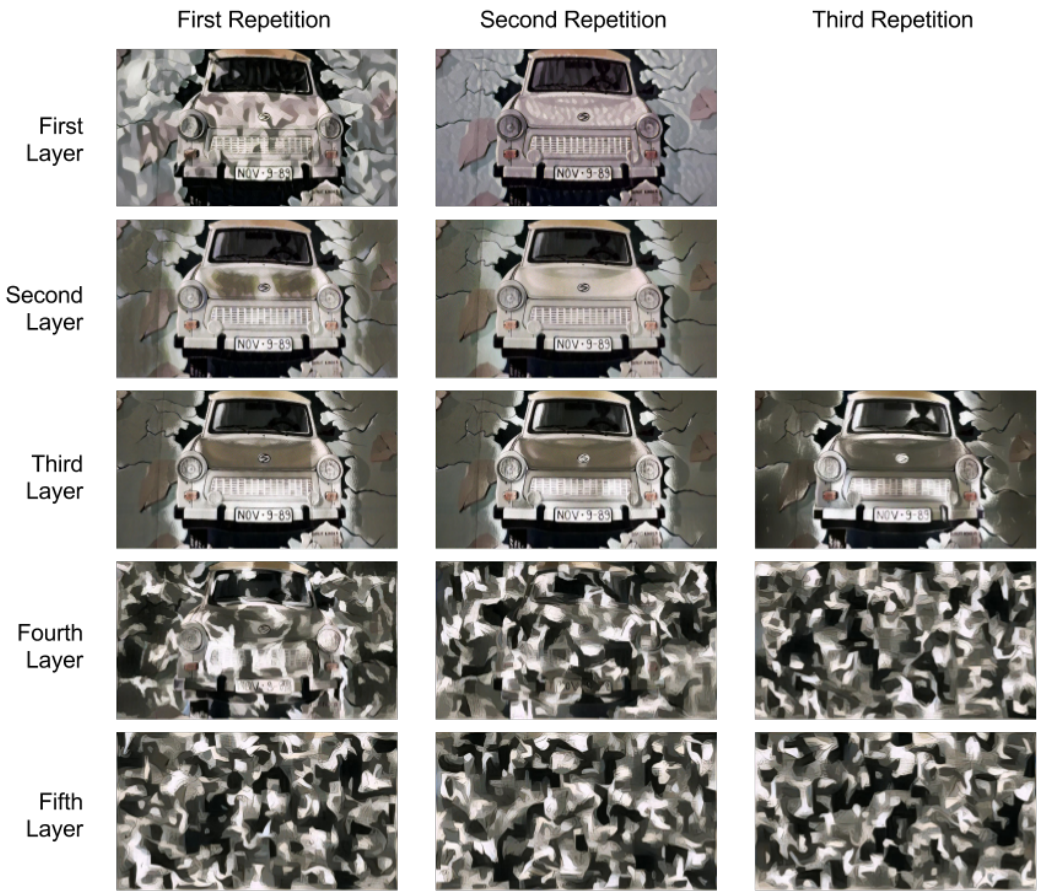

Figure 3.13: Content Representation using different convolutional layers and different repetitions

the higher the value of the ratio, the more style influence on the output. Using as content image Figure C.14 and style Figure C.7, in Figure 3.16 this difference can be observed, but with higher values, the results are very similar. As such, the ratio at 1×10^4 is adequate for generating images with a balances influence of style.

Thirdly, the selected initialisation image determines what is the starting point of the system. Just like artists start from content reconstruction by observation, this algorithm can use as initialisation the content image, as suggested in [Novak & Nikulin, 2016], instead of starting with a noise image, as suggested in [Gatys *et al.* , 2015]. If the initialisation is by a noise image, the result will be a pure combination of style and content, but it will require more iterations to obtain a visually appealing result. If the initialisation is by the content image, this means the system will change a replica of the content image and do the approximation to the transfer of style. Of course, by doing this the content characteristics are ensured and the influence of style is slightly reduced. Also, it is possible to initialise the system with the style image or a mixture of content and style[8] as suggested in [Fzliu, 2015]. The selection of the initialisation image will depend on

[8]It is important to highlight that the algorithm has three inputs, the content and the style image, as described before,

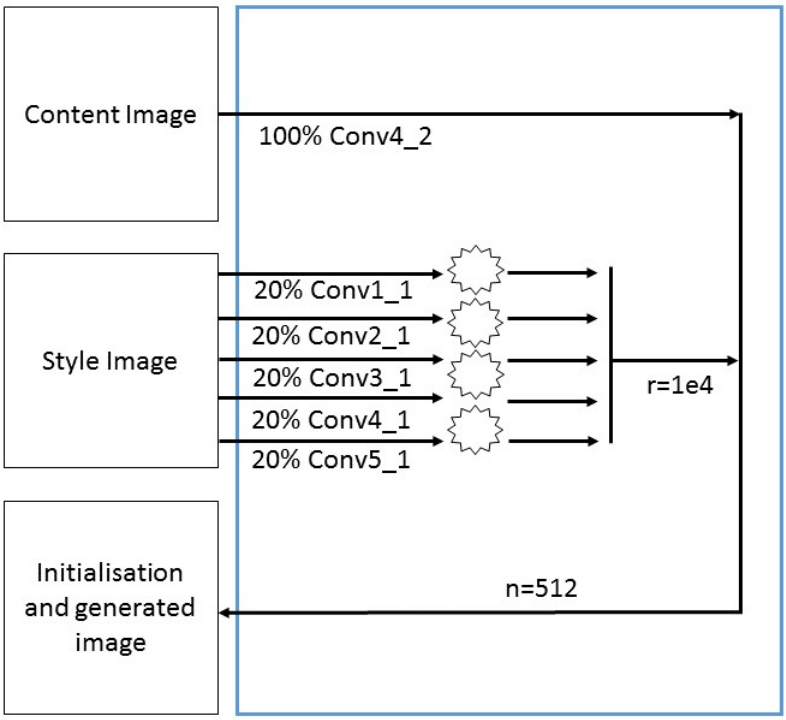

Figure 3.14: Scheme representing the algorithm developed by [Gatys *et al.* , 2015]: with conv4_2 as the convolutional layer representing the content and 5 convolutional layers for the correlations of style. The weight of the style component is represented by a rate, multiplied by the style loss component. The scheme presents an example of 512 iterations for the approximation of the generated image.

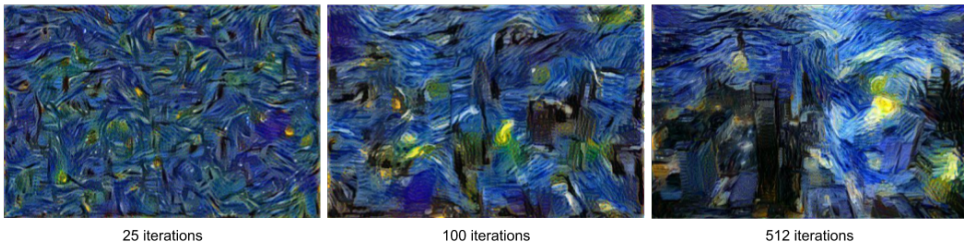

Figure 3.15: Results by varying the number of iterations

what is aimed to be achieved. In Figure 3.17, using as content image Figure C.15 and style Figure C.16, it is presented some results obtained by varying the initialisation image and the number of iterations. Comparing content and noise initialisation, although by starting with a noise image the system provides a more flexible result and the output will be more visually appealing, starting with the content image provides a good result with a lower number of iterations. This means the

and the initialisation image, that can be a noise image, a replica of the content image, a replica of the style image or a mixture of the content and style images.

Figure 3.16: Varying the value of the style ratio: 1×10^2, 1×10^4, 1×10^6 and 1×10^8

processing time will be inferior. Nevertheless, using as input for initialisation the content image will only be an acceptable solution if it is intended that the output has clear presence of content and if the artistic abstraction from the reality can be disregarded.

In Figure 3.18, using as content image Figure C.17 and style Figure C.3, it is presented a second example using a mixture of content and style as the initialisation image and comparing this to the standard initialisation with noise. It is remarkable the resemblance between the images obtained with 512 iterations and initialisation with a noise image and with 150 iterations starting with a mixture of content and style. As discussed above, with an increasing number of iterations, the output converges to a visually appealing combination between content and style. On the first case in Figure 3.18, although there is evidence of content, the number of repetitions does not seem to suffice to reach a good combination. To improve this, the number of iterations should be greater, as would be the processing time. On the second case, the output was processed with only 150 iterations, and resulted in a balanced combination of content and style. In this case, the advantages of choosing a mixed initialisation are notorious.

Finally, it is important to highlight that the selection of the Network model[9] can influence

[9]More about this in Appendix 2

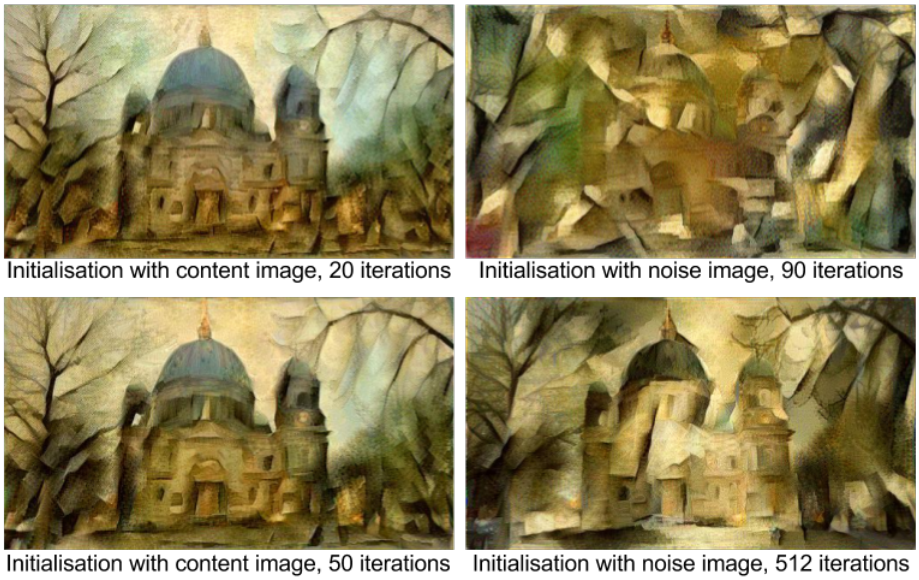

Figure 3.17: Varying the initialisation image

Figure 3.18: Varying the initialisation image

not only the time of execution, but also the final resulting image, as each model has a different number of layers and, consequently, different calculated weights. The difference between the results produced by three different models is shown in Figure 3.19, all of them generated with

50 iterations and the same values for the variables of the system. To produce these images, the selected content image can be seen in Figure C.10 and the style in Figure C.11.

Figure 3.19: Results obtained with different models of the Network

3.3 Conclusion and analysis of the selected approach

This chapter presented a short explanation of the selected style transfer technique and brings some empirical basis to what were the results obtained by the combinations of changes in these variables of the system and a comparison of outputs. In order to apply this style transfer system to 3D models, it is important to fully understand what are the results with the alteration of some of the parameters presented in Section 3.2 so that the system can be adapted to 3D style transfer with adequate control values, concerning the number of iterations, the reference style and content convolutional layers, the selected Network model and the initialisation image and the style ratio.

Considering the tests presented in this Chapter, the layers selected for content and style comparisons should coincide with what is presented in Table 3.1 because these were the ones that produced more balanced and visually appealing results. An adequate value for the style ratio would be 1×10^4 to obtain images with a strong influence of style. The initialisation image can vary between a noise image, the content image or a mixture of the content and style images and any of these options influences the number of iterations required to reach an optimal solution, concerning the minimisation of the loss function, as does the selected Network model. The number of iterations should have an adequate value for the results aimed to be achieved.

The analysed code resulted of the article of Gatys et al. released in 2015 [Gatys *et al.* , 2016], but a new version was referenced in 2016 by the same authors [Gatys *et al.* , 2016] where a suggestion for colour control was described, in order to enable the generation of images loyal to the colours of the content image, instead of the style. This version of the algorithm can be relevant for this dissertation and will be used as a comparison to the results obtained by different techniques in the next chapters, but was not fully analysed in this chapter because the code is based on the same main structure, only adding a few constraints.

Chapter 4

Artistic Style Transfer for textured 3D models

4.1 Objectives and Problem definition

The main goal of this dissertation is to transfer any selected style from a 2D representation of an art work (i.e. an image) to a 3D model described by an object file and its texture map. This method should not depend on any kind of 3D software for its implementation and there should not be any kind of restriction in the image chosen for the representation of style. The style is described as colour and texture, discarding any kind of figure deformation to the 3D model. For this application, any user can be the artist that stylises the model, i.e. it is not required artistic knowledge for the matter. It is intended to achieve as results visually appealing stylised 3D models.

After analysing how the algorithm for Style Transfer works [Gatys *et al.* , 2015] in 2D transference, this chapter focuses on how to apply this same algorithm in a 3D model, more specifically transferring the style from one image to the texture map of a 3D model. As a texture map is not exactly a simple 2D image, by applying the style in the same way as before, it is expected that there are some problems related to the geometrical positions of the texture islands [1] in the image. To solve this, a constraint should be added to the original algorithm of style transfer as a spatial control of the output. This constraint should be more effective by changing the values of some parameters of the system, namely the ratios of the loss functions and the number of iterations.

4.2 Style Transfer applied to 3D models

As explained in Chapter 2, the texture of a 3D model is mapped from a 2D image, by corresponding 2D coordinates to the 3D vertices of the model. In Figures 4.1 and 4.2 show two examples of

[1] The texture islands of a texture map are isolated areas that correspond to pieces of the 3D model. They are mapped to be adapted to the model.

3D models, processed by the software Meshlab [Cignoni *et al.* , 2008] [2]. A texture map is the representation of the 3D texture in a 2D image, and this correspondence is shown in Figures 4.3 and 4.4. So, if one intended to stylise a 3D model, in a first glimpse, it would be intuitive to try to apply the style of a painting directly to the texture map.

Figure 4.1: 3D model of a mandrill

Figure 4.2: 3D model of a shark

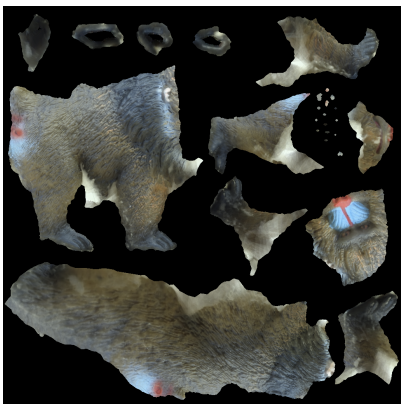

Figure 4.3

Figure 4.4: Shark model 4.2 texture map

Using the algorithm of [Gatys *et al.* , 2015], the application of style to a 3D model should be a simple transference of a style from a chosen painting to a texture map like shown in Figures 4.3 or 4.4, as it was explained in 3. For this task, the system uses as the content image a texture map, and as the style image a painting of choice, which in this case was The Scream, by Edvard Munch, shown in Figure C.3. The output of the system is a 2D image with the content from the texture map and the style applied, as shown in Figures 4.6 and 4.7. The results of this first test can be observed in the images of the 3D models, in Figure 4.5, .

[2]Meshlab is an open source 3D mesh processing system and was selected as the software to compare the 3D renderings during the testing process of this dissertation. All the images representing 3D models in this document were processed with Meshlab, unless indicated otherwise

Figure 4.5: Results of applying a style to a texture map, using the algorithm of [Gatys *et al.* , 2015]

4.3 Discontinuities

To generate the images presented in Figure 4.5, the style of painting C.3 was transferred using the algorithm [Fzliu, 2015] based on [Gatys *et al.* , 2015] algorithm. In short, the style was applied directly to the texture maps presented in Figure 4.3 and 4.4, resulting in the Figures 4.6 and 4.7 as an output of the style transfer system. After this, the new texture maps are applied to the 3D models.

Although the selected style has a strong influence on the result, if observed in close detail, there is an evident flaw in these representations, which will be referred as texture discontinuities in this document.

Texture discontinuities are the areas where the progression of the texture is disrupted, revealing an evident break between the islands of the texture maps. The evidence of these discontinuities is

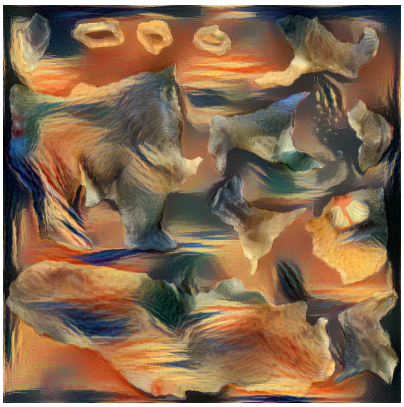

Figure 4.6: Mandrill texture map 4.3 stylised

Figure 4.7: Shark texture map 4.4 stylised

highlighted in figure 4.8 [3]. In this first approach to apply a selected style to a 3D model, there are some considerations worth to mention for further understanding the next chapters. First of all, the results obtained by the simple transference of style to the texture maps are not satisfactory, and so a plan for a solution concerning this will be presented in the next section. Second of all, these results correspond to what was predicted, considering the distance between neighbouring islands. This is why the solution will aim at having a spatial constraint to connect the points that ought to be neighbours in the 3D model representation. Lastly, by applying the style directly to the texture map, constraints concerning the selected softwares for the 3D renderings are avoided, enabling the user to manipulate the texture transfer independently from the choices over what is the software used to manipulate the 3D model.

Although the models originally have a texture that is smooth along the whole object, when applying a style to that texture, the discontinuity is evident where two distinct islands are put together. As shown in figures 4.3 and 4.4, a texture map can be composed by several islands that will be connected in the 3D model, but in a 2D representation, they lay in different regions of the image. The style transfer algorithm proposed by Gatys et al. [Gatys *et al.* , 2015] applies the style to a 2D image, taking the pixel neighbouring information into account, but not the connectivity across spatially separated texture islands, which would be important in this case. But even if it is considered a texture map without multiple islands, as it would be a cube plan, that represents a perfect unwrapping of the figure, distant edges in that texture map are connected side by side in a 3D model of a cube. So the problem of discontinuity when applying a style remains.

4.4 Style Transfer applied to 3D models with border constraint

Based on the method developed by Gatys et al. [Gatys *et al.* , 2015] explained in Chapter 3, the proposed solution to avoid discontinuities in the 3D style transfer is based on adding a spatial constraint, similar to the approach of adding time constraints as purposed in [Ruder *et al.* , 2016b], by

[3]The representation of discontinuities is in figure 4.5 and can be compared to the images in figure 4.8, with the critical areas highlighted in green

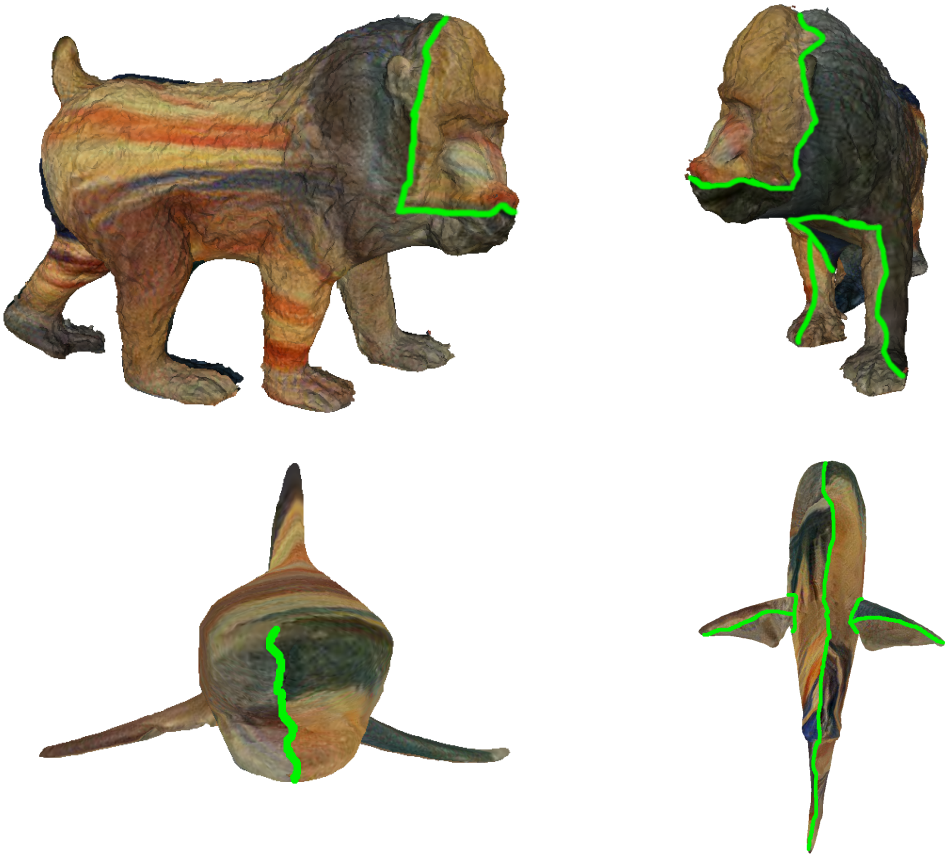

Figure 4.8: Evidence of texture discontinuities

maintaining pixel information as input of the system. In this case, the problem relies on the pixels belonging to the borders of the texture islands. A texture map is a 2D representation of a texture from a 3D model. It can be organised in several ways, and so for a single model, various combinations of texture maps can be generated, with different number of texture islands [4]. One could assume that if the texture map is formed by a single island, as an unwrapping representation of a figure, the problem would not remain. Nevertheless, the discontinuities are between distant pixels in the 2D representation that are connected as neighbours in the 3D model, and so the problem resides regardless of the number of islands, although this influences the level of discontinuity. All of the pixels on the border of the island have a neighbouring connection in 3D that is not represented in the 2D texture space. Increasing the number of islands means there are more pixels with this same correspondence problem. When applying a style, the style transfer algorithm analysis the colour RGB values of the texture by layers and adapts the pixel values of the generated image to the feature response correlations, as explained in Chapter 3. This results in an homogeneous representation of style in the output, because in a common 2D image, the neighbouring pixels are actually next to each other spatially. A texture map is not a common 2D image because, as

[4] In this document, a texture island is defined by an isolated region of texture in the texture map, as it can be observed in figure 4.4, with 3 texture islands

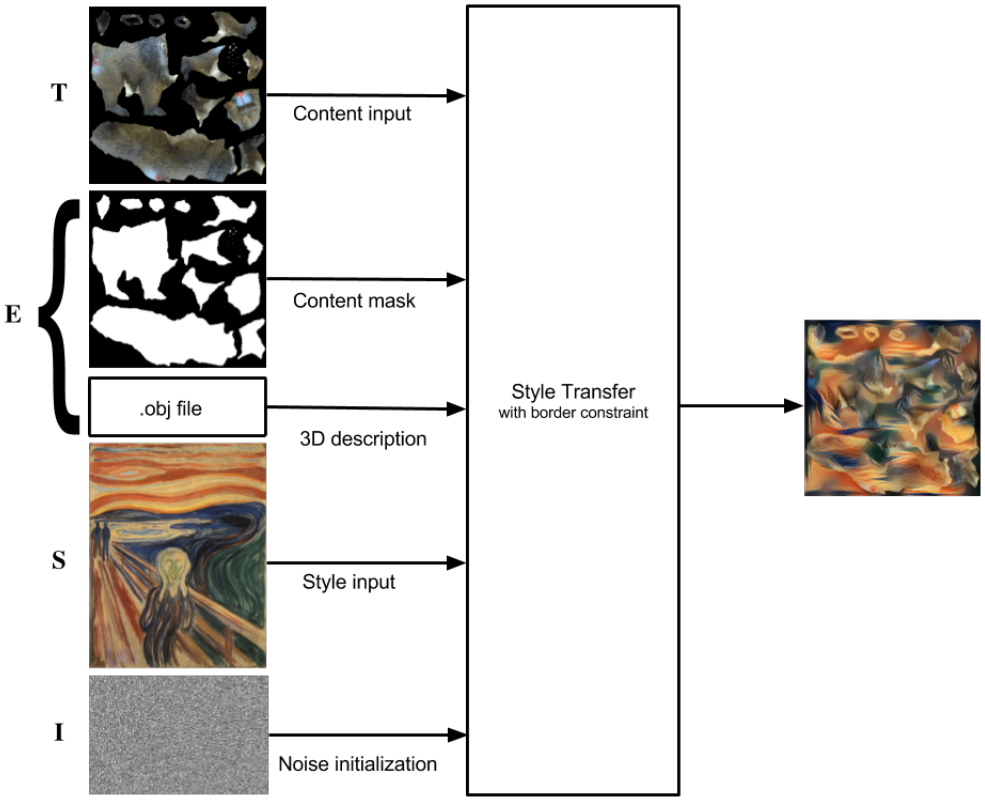

Figure 4.9: Representation of the proposed solution

the name implies, it is a mapping representation. This is where the style transference fails, as an homogeneous transference will provoke discontinuities when the output is applied as texture of the 3D model. And so, the presented solution for this matter focuses on giving the information of the texture map to the style transfer algorithm, so that the the neighbouring pixels can be clearly identified and it can be guaranteed that they share the same style application.

$$\mathscr{L}(I,T,S,E)_{total} = \alpha \cdot \mathscr{L}(I,T)_{content} + \beta \cdot \mathscr{L}(I,S)_{style} + \mu \cdot \mathscr{L}(I,E)_{borders} \tag{4.1}$$

As it was explained in Chapter 3, style can be considered as a combination of colours, light and texture, and the constraint described in this section considers that the convergence to equalised neighbouring pixels is achieved by pixel colour correspondences. This is easily translated in Equation 4.1, where I is the generated texture map, T is the original texture map, S is the style image and E is the pixel correspondence information. This Equation continues the total loss function defined by Gatys et al. [Gatys *et al.* , 2015] but with the parameter of the border loss added. Each of the losses has a considered weight, represented by the variables α,β and μ, that indicate the influence of each part in the output. The total loss function has the two functions of content loss $\mathscr{L}_{content}$ and style loss $\mathscr{L}_{style}$ that depend on the CNN and are optimised by means of the back-propagation step, and the $\mathscr{L}_{borders}$ parameter, that directly depends on the values of the generated image in each iteration and is independent of the CNN structure.

The scheme of the new system is shown in Figure 4.9, with two more inputs to the style transfer algorithm: the object file (.obj) that contains the list of 3D vertices, texture coordinates, and their correspondences, and the mask of the texture map, used to define the borders of the texture islands. The texture map is the content representation, since it is the image to where the style will be transferred.

Having a noise image of size $[height \times width \times channel]$[5] that serves as input for initialisation of this system, it will be changed repeatedly in every iteration for a balanced combination of both content and style. This image is also know as the generated image, because it starts as an input but it changes and converges, becoming the output in every iteration. The content will be based on the comparison between the feature responses in the layers of the CNN, and the style only on the correlations between feature maps, just as before. The generated image will have a combination of content and style, but in the regions defined as borders of the content image, i.e. the borders of the texture islands, the pixels will be compared in order to align the values of neighbouring regions in the 3D models, that are not spatially side by side in the 2D representation. The borders of the texture islands can be related to vertices in the 3D model that appear twice in the 2D texture space due to the cuts involved in the 2D-3D mapping, from object surface space to texture space. A connection (i.e. and edge) between these vertices can be established via the surface triangulation. Thus, boundary corespondent texture island pixels can be addressed by these vertex connections. So, given a list of corresponding edges represented by the starting and ending points in texture space coordinates, they are aligned by reducing the difference between them. So for an island A that will be next to an island B, the borders that will be put together in the 3D model are defined by a list of corresponding edges e_1^i, e_2^i where i is the index across all edges that are included in the texture boundaries, assuming e_1^i is part of the boundary of island A, and e_2^i, from the island B. Although the pixels covering those edges are not next to one another in the texture map, or later on in the generated image, it is known by the information contained in the object file (i.e. .obj, also serves as an input to the system, as shown in Figure 4.9), that these pixels should be equal because they represent the same connection in the model. Accordingly, it is purposed an additional term that penalises the difference between the pixel values contained in these two edges, so that when they are put together, there will not be a discontinuity between the islands A and B, producing a coherent style application in this region. This border penalty is described by Equation 4.3.

$$\begin{aligned} e_1(\lambda) &= \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + \lambda \left[\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} - \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \right] \\ e_2(\lambda) &= \begin{pmatrix} x_3 \\ y_3 \end{pmatrix} + \lambda \left[\begin{pmatrix} x_4 \\ y_4 \end{pmatrix} - \begin{pmatrix} x_3 \\ y_3 \end{pmatrix} \right] \end{aligned} \tag{4.2}$$

$$\mathscr{L}(I,E)_{border} = \sum_i \sum_\lambda \frac{1}{2} \left(I(\lfloor e_1^i(\lambda) \rfloor) - I(\lfloor e_2^i(\lambda) \rfloor) \right)^2 \tag{4.3}$$

[5]The channel could be equal to 3 if it is an RGB image

$$\lambda = \frac{n}{a}\ ,\ n = \{1,2,3,...,a\}$$
$$a = max\ \{||e1||, ||e2||\} \tag{4.4}$$

The Equations 4.3 and 4.2 represent the difference between the pixel values contained on the texture islands edges. The border penalty representing the spatial constraint is minimised in every iteration, so that for every pair of corresponding edge indexed by $i \in \mathbb{N}$, the distance between the corresponding pixel values are as small as possible to avoid texture discontinuities. As the edges e_1 and e_2 may have different lengths, because there is some deformation when projecting the texture in 2D, the λ parameter represents an equidistant sampling of the edge such that every pixel is covered. According to Equation 4.4, λ is related to the size of the longer edge, in order to have a value correspondence for every pixel of both edges. E.g. if the edge e_1 is of length equal to four pixels, and edge e_2 of only two pixels, the variable $a \in \mathbb{R}$,representing the maximum edge length, is given by $a = 4$, and λ will be in the range $[0,1]$ with step size of 1/4, This means that it will assume 4 values for edge e_1 approximated to integer values by a floor function, to make a correspondence to pixel values. The same would happen for edge e_2, approximating to only two pixel values, considering that is the length of this edge. This explanation is illustrated by Figure 4.10.

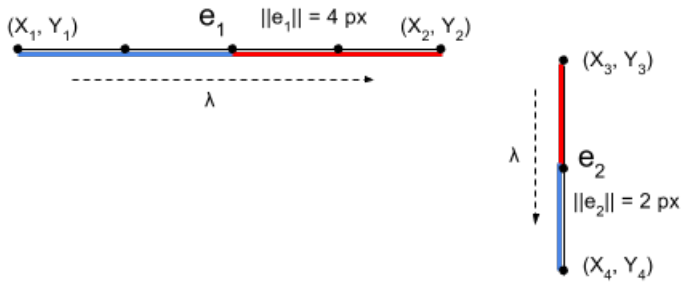

Figure 4.10: Representation of the edge correspondence

In order to do the minimisation of the total loss function $\mathscr{L}_{total}$, the pixel values of the generated image are adjusted in each iteration towards the minimisation. The minimisation process requires the gradient of the function, so that the values can be changed in the right direction and amount, and this step of the minimisation of the loss function relies on the backpropagation task of the Network to calculate the dependencies of the layers to the initialisation image pixel values. By changing the pixels in every iteration to reduce the difference between the generated image and the style and content image, this step only ends when the optimal solution is achieved or when the number of iterations is reached. As the parameter of the border loss $\mathscr{L}_{border}$ does not depend on the feature responses, calculating its gradient does not require the backpropagation step. The gradient will be with respect to the generated image and is generalised by the Jacobian matrix, as it

is a function of multiple variables. The Jacobian is of size [height x width x channel], representing the (x) pixels in the generated image, and it is described in equation 4.5.

$$J(x) = \begin{cases} I(\lfloor e_1^i(\lambda)\rfloor) - I(\lfloor e_2^i(\lambda)\rfloor)\,, & x \in \lfloor e_1^i(\lambda)\rfloor \\ -(I(\lfloor e_1^i(\lambda)\rfloor) - I(\lfloor e_2^i(\lambda)\rfloor))\,, & x \in \lfloor e_2^i(\lambda)\rfloor \\ 0\,, & otherwise \end{cases} \tag{4.5}$$

Having equalised the pixel values belonging to the edges, it is expected that with further iterations, the colour equality from corresponding border pixels would fuse spatially to neighbouring pixels due to the convolutional kernels.

4.5 Implementation of the border constraint

The solution explained in the previous section is an additional spatial constraint to the style transfer algorithm [Gatys *et al.* , 2015]. As such, for the implementation, the code used for the initial style transfer system was [Fzliu, 2015], adding the border loss function and its gradient, as well as the new inputs required for this new constraint (i.e. and object file and a texture map mask).

Starting by the simplest model tested, the cube has a list of fourteen edges in its texture island borders, and so by producing a list of edge correspondences, the result is shown in Table 4.1, with seven pairs of neighbouring edges. This list contains the starting and ending point and this serves as an input to the border loss function[6], where e_1, e_0 represent the corresponding edges as exemplified in Figure 4.10, with (x_1, y_1) and (x_3, y_3) as starting points and (x_2, y_2) and (x_4, y_4) as ending points. In this case, i represents the total number of edge pairs. As the cube has 7 neighbouring sides, $i \in [0, 6]$.

Table 4.1: Cube edge pairs

$e_1, e_0, i = 0$	(208, 0)	(208,128)	(80,256)	(80,384)
$e_1, e_0, i = 1$	(463, 384)	(335, 384)	(335, 384)	(335, 512)
$e_1, e_0, i = 2$	(335, 128)	(336, 256)	(463, 256)	(336, 256)
$e_1, e_0, i = 3$	(208, 512)	(335, 512)	(208, 0)	(336, 0)
$e_1, e_0, i = 4$	(208, 256)	(80, 256)	(208, 128)	(208, 256)
$e_1, e_0, i = 5$	(335, 128)	(336, 0)	(463, 384)	(463, 256)
$e_1, e_0, i = 6$	(80, 384)	(208, 384)	(208, 384)	(208, 512)

To calculate the distance between the edges, for each pair, the length of both edges is compared, to define the value of $a \in \mathbb{R}$. With this, the value of λ is known. After reading the list of edges, for each pair, the maximum length is determined and the border loss function is calculated in every iteration as shown in Equation 4.3 and the evolution of the loss comparing to the content and style losses can be visualised in Figure 4.12. This graphic represents the values of each part of the loss function so it is possible to compare the evidence of each loss in the minimisation.

[6] The considered image has a dimension of 512x512, i.e. the texture map

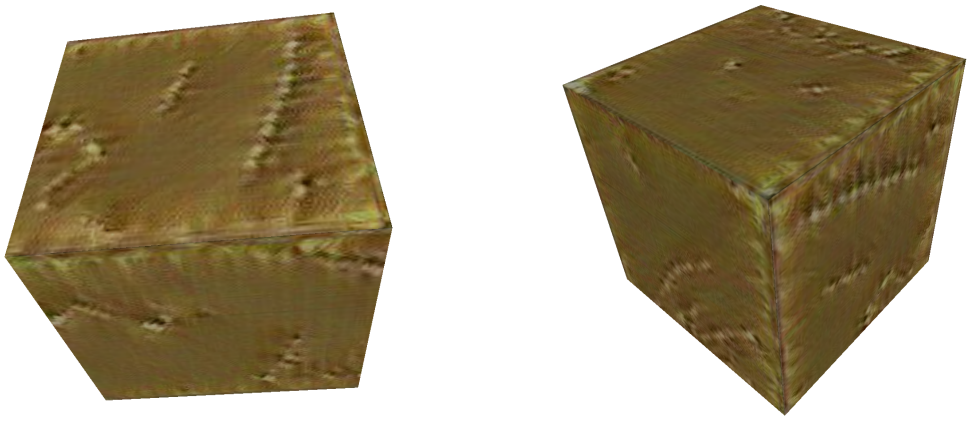

Figure 4.11: Results of stylised cube model with border constraint

For each pair of edges, the reduction of the border loss is shown in Figure 4.13, and it represents the mean value of the loss for a varying λ, and the sum of the colour components RGB, multiplied by μ. The results of this test are presented in Figure 4.11, with a stylised texture produced with only 18 iterations. Here it is visible that the zero value is not achieved until the iterations are completed, so the optimal value is not reached.

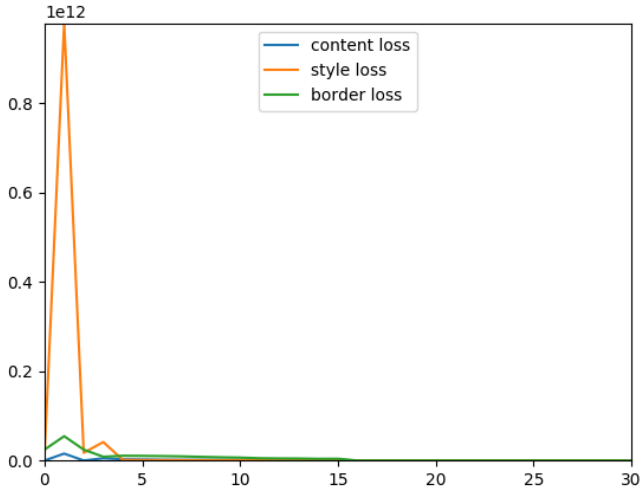

Figure 4.12: Reduction of loss functions, evolution with iterations. The graphic has as amplitude the values of the losses, with units in the range of 1×10^{12}. The x values represent the 18 iterations.

As it was explained in Chapter 3, the original system has several variables that enable some control on the output, one of which is the number of iterations. As before, the best results should be achieved with more iterations of the system because the generated image converges to a better balance between content and style, as the content and style loss functions are minimised. With this new spatial constraint, more iterations provide better possibilities of reduction of the border loss function, and so the difference between edges is expected to reduce as well.

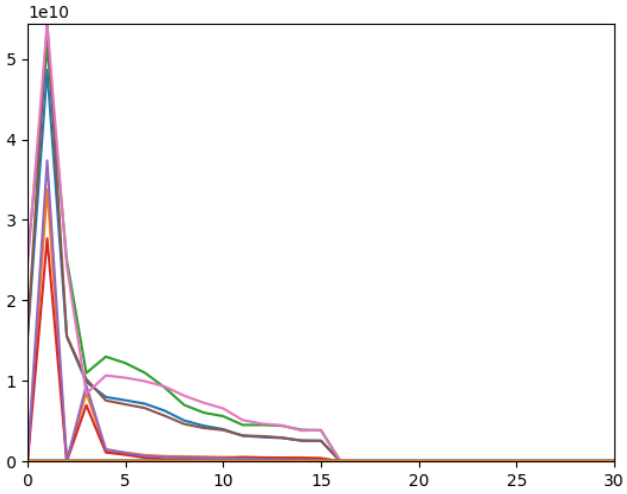

Figure 4.13: Reduction of distance between neighbouring edges, evolution with iterations, with 10 iterations and $\mu = 10^6$. The graphic has as amplitude the values of the losses per edge (identified in different colours), with units in the range of 1×10^{10}. The x values represent the 18 iterations.

Another set of results is presented in Figure 4.14 where the model selected was a representation of a shark. The list of edges is very long, as it is a much more complex model than the cube presented before.

Figure 4.14: Results of stylised shark model with border constraint, with 18 iterations and $\mu = 10^6$

In a detailed observation of the results for both the cube and the shark, it is still very evident the discontinuities between the texture islands. In these cases, the style transfer system only used fifteen iterations to produce the stylised texture maps. Also, as it was explained in the previous section, a weight value is multiplied by the border loss as a variable to control the influence of this in the final result. For this set of results, the weight value was $\mu = 10^6$.

It can be observed in Figure 4.12 that the weight of the border loss function is of less importance than the content and style loss functions, and so, the importance of the spatial constraint is not considerably relevant in the examples shown. As such, after these experiments the value of this weight was increased, as was the number of iterations.

The new results can be observed in Figures 4.15, 4.16 and 4.17. Here, the output converged during 48 iterations and the weight had a value of $\mu = 10^8$.

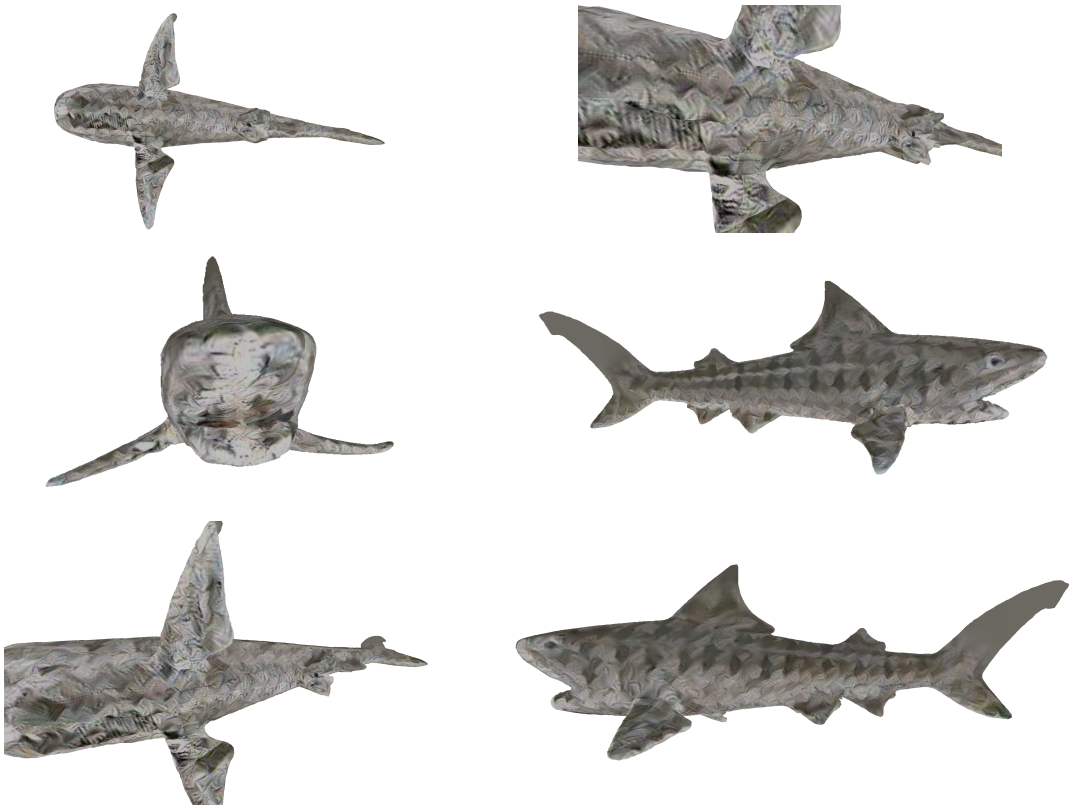

Figure 4.15: Style transfer with border constraint, results obtained with the method purposed in this dissertation. Shark model with 48 iterations

Table 4.2: Example values of border loss

$e_2, e_1, firstiteration$	5.12×10^9
$e_2, e_1, lastiteration$	4.53×10^9
$e_3, e_4, firstiteration$	18.86×10^{11}
$e_3, e_4, lastiteration$	2.44×10^{11}

Again,the discontinuities are still visible between the texture islands, and in the graphics presented in figures 4.16 and 4.17, the final values (i.e. in the last iteration) of the total border loss and the loss per edges are not zero, and so the minimisation was not completely successful. Also, the evidence of the border loss comparing to the other parameters of the total loss function is not so emphasised. As a result, the visualisation of the style in the images presented of the 3D model have still evidence of discontinuities, which is in accordance to the values shown in the

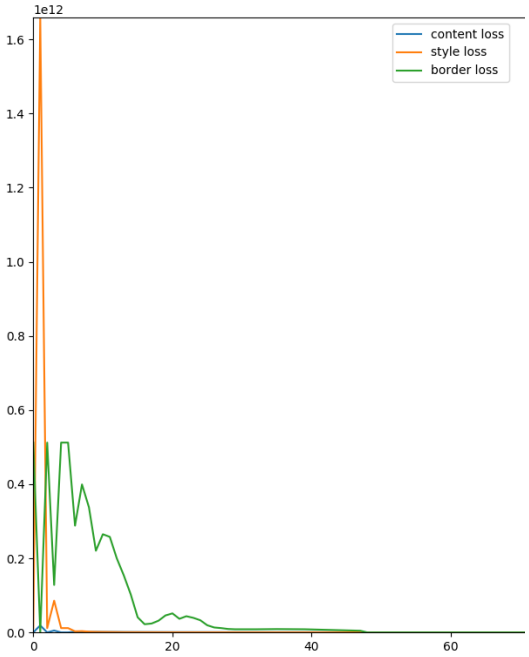

Figure 4.16: Reduction of loss functions, evolution with 48 iterations and $\mu = 10^8$. The graphic has as amplitude the values of the losses, with units in the range of 1×10^{12}. The x values represent the iterations.

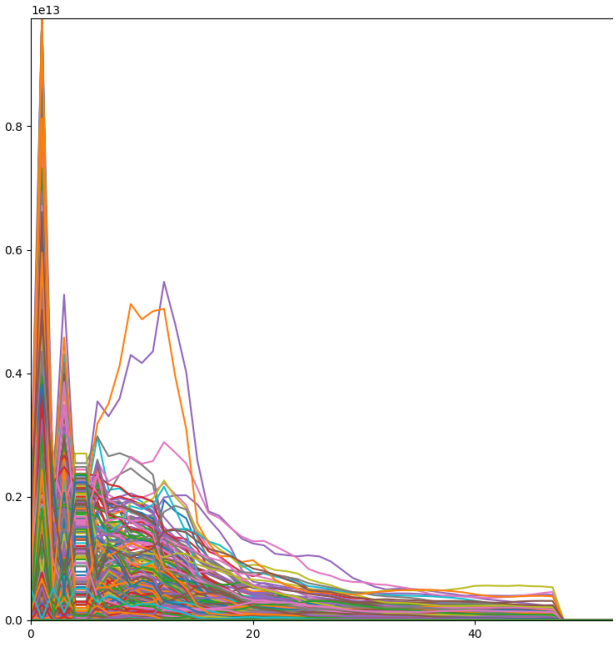

Figure 4.17: Reduction of distance between neighbouring edges, evolution with iterations, with 48 iterations and $\mu = 10^8$. The graphic has as amplitude the values of the losses per edge (identified in different colours), with units in the range of 1×10^{13}. The x values represent the iterations.

graphics. However, these Figures show that the values are drastically decreased in each iteration, demonstrating a great reduction on the border loss function, as it was intended. This can also be concluded from the example values in the Table 4.2, presenting the initial border loss of two pairs of edges, and the loss value of the same pairs in the last iteration. In this case, these values were taken form the test presented in Figure 4.15, and they were calculated as the mean value of the loss for a varying λ in each edge, summing the colour components RGB, multiplied by μ.

4.6 Increasing the weight of the Border Loss

As it was explained before, the minimisation of the content loss and style loss parameters take into account the weights considered for each part, identified as α and β, as show in Equation 4.1. Considering this, the solution described in the previous Section was modified in order to emphasise the border loss parameter, so that it has a stronger influence on the generated image. This is achieved by multiplying the μ ratio by the border loss parameter, in order to maximise the influence of this in the minimisation process, since the relevance of this parameter did not have an adequate weight in contrast to the content and style parameters.

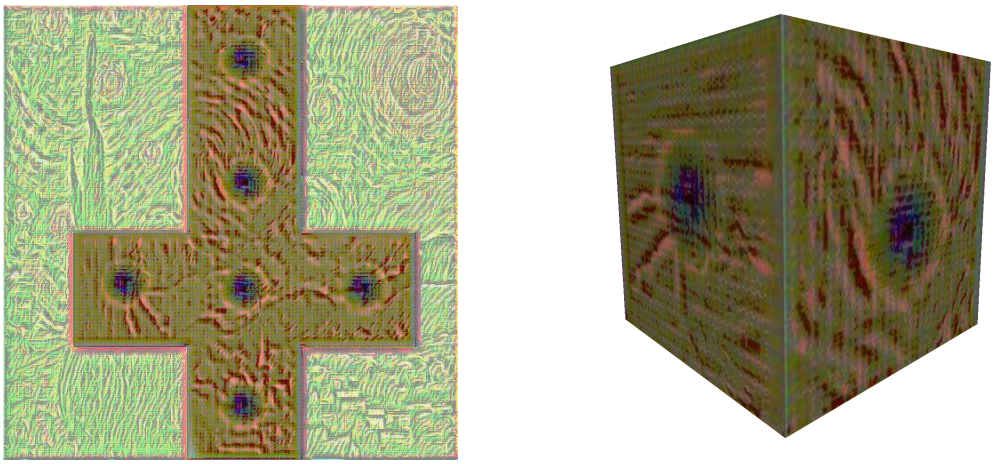

Figure 4.18: Results of stylised cube model with evident border constraint

In Figure 4.18 it is shown the resulting texture map and a view of the cube 3D model obtained with this change in the implementation of the solution presented. These were obtained with a low number of iterations, using the GoogLeNet[7] model for the Network and with $\mu = 10^7$ and using Figure C.12 as the input style.

Here, there is a strong evidence of colour equality in the borders of the cube, and so the border constraint showed a more evident presence in the result, as it was expected.

With a deep weight in the border loss, the minimisation step highlights this constraint, as its gradient will show not only the direction in which the pixels of the borders should be changed, but also the amount that should be altered, i.e. how much is changed in every iteration.

[7] See appendix 2

4.7 Obtained results

In this dissertation, to transfer style to a 3D model, three methods were applied, and the comparison between the contained results is described in this section.

Firstly, by applying the original algorithm of [Gatys *et al.* , 2015] to the texture maps, a problem of discontinuity was detected. The results were visually appealing because they had a great influence of style, but there was not a smooth transition of style between texture islands. More results on this are shown in Figure 4.19.

To solve this, a method was proposed based on adding a spatial constraint to the same algorithm. Despite producing results that register a relevant reduction on the border loss, the models still had influence of these discontinuities. Results are presented in the previous sections, namely in Figure 4.15 and 4.18.

Lastly, as stated in Chapter 2, the same authors released a new algorithm with more controlling parameters to the system. One of these is with respect of maintaining the original colour of the content image. As such, in this section there are presented some results of the transference of style using this technique, shown in Figure 4.20. Here it is still evident the discontinuity on transitions between islands.

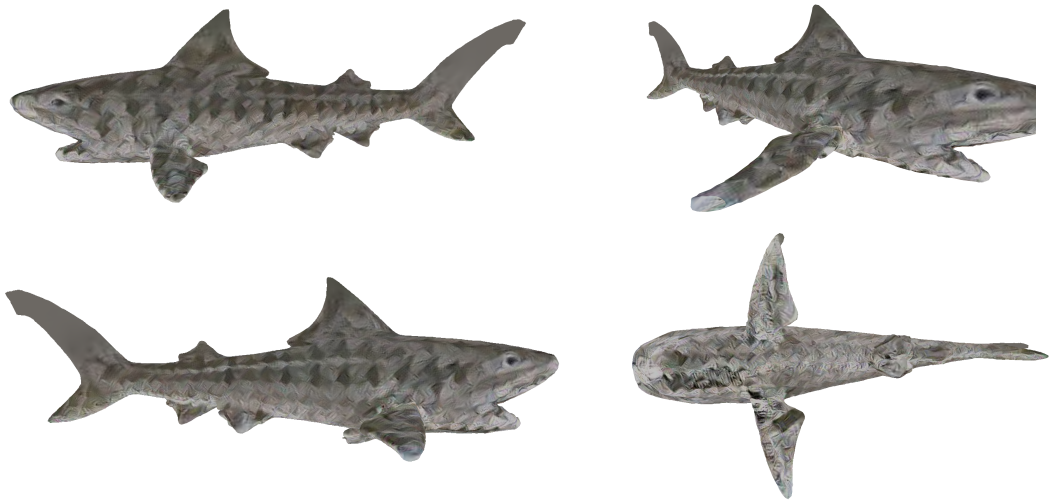

Figure 4.19: Original style transfer method

Comparing the two methods in Figures 4.19 and 4.20, one can observe that keeping the original colour of the content image may produce results with slightly less evidence of the discontinuities. By discarding the colours from the style, the discontinuities are not so evident but the results are less visually appealing. With the original algorithm, the results have a much stronger evidence of style, as the colours are kept from the original style image. Nevertheless, the problem of discontinuities is also more evident, as in both of these systems, the style is applied to the 2D image based on the same method.

Figure 4.20: Style transfer for light component maintaining original colours of content

This dissertation presents a spatial constraint intended to solve these discontinuity problems between texture islands, but in the images presented of the 3D models, it is still possible to perceive signs of discontinuity, demanding for smoother transition of style between texture islands. The solution presented in this dissertation did not show the results expected concerning the discontinuities between texture islands. Nevertheless, the solution of adding a spatial constraint has very promising results in the reduction of the border loss function, and these has proven to be a first step for solving the problem of discontinuity.

The border constraint solution was not completely effective in the results presented in this Chapter, but this is most certainly related to the parameters chosen for controlling the system. This means that by not only increasing the weight of the border loss function in order to become a strong parameter in the minimisation process, by also increasing the number of iterations for the system convergence, the results can reflect the decreasing values expected in the minimisation function, as it was depicted in the graphics presented in the previous sections.

All in all, this chapter presents the implementation of the main objective of this dissertation, that was transferring a style from a 2D image to a 3D model. However, during this process, some difficulties were revealed and these are still a problem to the 3D style transfer.

Chapter 5

Conclusion and Future work

5.1 Achievement of objectives

The focus of this dissertation was on the development of a method to apply artistic styles to textured three-dimensional models. Firstly, a collection of research on the topic of style transfer was collected in order to choose the best approach to implement a solution for the problem. Secondly, the selected approach was deeply studied and analysed. Also, an analysis of this style transfer matter was presented in a more artistic perspective, in order to clarify what is style and what can be achieved with deep style transfer. After selecting the algorithm for 2D style transfer, the implementation for style transfer to 3D models started by using the 2D system and applying it in texture maps. The results were visually appealing considering the strong influence of style in the models. However, a problem related to the spatial organisation of the texture map revealed discontinuity problems between texture islands, preventing a smooth appliance of style to the models. In order to solve this difficulty, a method with a spatial constraint was developed as the purposed solution. Actually, other approaches were taken into account, but the introduction of the spatial constraint based on border loss reduction has proved to be the most effective approach for solving the problem. The model developed by [Gatys *et al.* , 2015] was used as a starting point for the resolution of these discontinuities problems, but further developments have been introduced namely by adding a border loss parameter to the initial equations.

5.2 Conclusion

With this dissertation, one can conclude that it is possible to transfer a style from a 2D image into a 3D model, by means of a Deep Style Transfer algorithm. This could be analytically confirmed by the graphics on the loss function reduction for increasing number of iterations, which obviously converge to zero but have not reached a zero neighbour interval that may prevent visual discontinuities in the output 3D models. Additionally, convergence for such an interval appears to be asymptotic while initial iterations yield drastic reductions and further successive reductions claim for a substantial increase of iterations.

On a first approach, the proposed solution did not solve the discontinuity problem described in Chapter 4 up to the point of the zero value in the loss function. A second approach was presented with better results, yet still not showing an absolute solution for the problem. However, results are very promising and this method is an important first step in producing visually appealing stylised 3D models. Expectations are that with a higher number of iterations of the system and increasing the weight of this constraint (so that this constraint is strong), the method developed in the scope of this dissertation can reach better results concerning the discontinuity problem, namely by reaching closer values to zero of the border loss function, which represents the difference between the edges of neighbouring borders of the texture islands. Having equalised the pixel values belonging to the edges, it is expected that with further iterations, the colour equality from corresponding border pixels will fuse spatially to neighbouring pixels due to the convolutional kernels.

5.3 Future work

In addition to the suggestions considering the improvement of the method presented in this document, namely by increasing the number of iterations combined with increasing the weight of the border constraint, there are other considerations for possible opportunities of research related to the topic.

Adding to the approach presented in this dissertation, the approximation of the borders could be expanded to the neighbouring pixels, instead of only equalising the bordering edges. Also, by separating the islands and applying the style individually, the influence between the content of different islands can be avoided. If the results accomplished by this solution are no satisfactory, an initialisation parameter can be considered, so that each style appliance starts with an island that has been already altered, i.e. by implementing the algorithm in a cascade mode: applying style to a first island, transferring the pixels in the borders to the neighbouring island and only then apply the style to this second island.

Other solutions for this style transfer technique can be considered. In the research of [Fišer *et al.* , 2016], it is presented a solution based on light correspondence of a 3D scene. The solution presented on this dissertation is based on texture colour transference, and it would be interesting to consider a correlation of the style to the light exposure, as artistically, this is a very important factor considering the style of a work of art.

In an artistic perspective, if one would consider the style of an artist as a combination of characteristic techniques, the style transference should be based on a combination of several works of art, instead of only one image. This was suggested by [Ikuta *et al.* , 2016], and could be applied in 3D models as well, to generate an object with a representative style of an art movement, for instance.

In [Gatys *et al.* , 2016], it is presented as an improvement a colour control parameter so that the colour of the original content can be kept. Although in an artistic perspective this does not meet the goals of transferring a style from an artist, because colours are usually part of a work of

art identity, it could be applied to 3D models if the purpose would be to maintain the main realistic characteristics of a model.

The method developed by [Selim *et al.* , 2016] can be very relevant for application of style to a facial 3D model, and this could be seen as a new research concern, specifically for facial artistic 3D representation. This method has a main concern of maintaining the facial expression of the content image, what could be achieved also by using specifically pre-trained facial recognition networks for the style transfer technique.

Other applications for this 3D style transfer algorithm can be considered, as a documenting system described in Chapter 1, to develop systems for visual documentations, helping patients to identify a disease, farmers to detect problems in the leaves of a plant or mechanics identify problems in materials or instruments.

Another interesting application for the 3D models style transfer could take advantage of the 3D environment and recreate spaces with a style representation, forming 3D paintings. For instance, there could be a 3D recreation of the Van Gogh's room C.8, not only with the colours, main forms and types of strokes, but also representing the perspective illusions that are present on the painting. Another example could be a recreation of a cubist space, based on a painting, e.g. **??**.

Appendix A

Style Transfer System

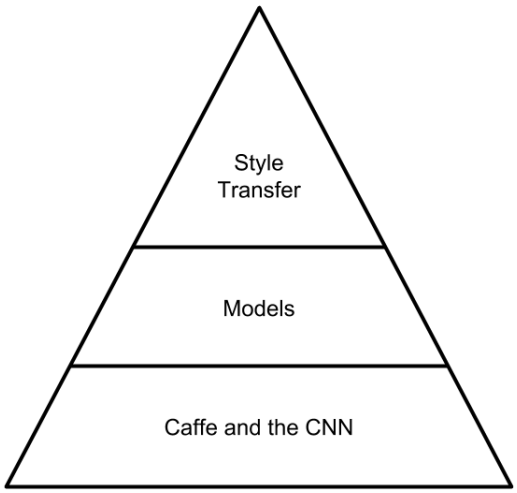

Figure A.1: Style Transfer System Representation

A.1 Introduction

For years the main concern for technological development has been to create artificial systems that would help the human being in difficult tasks: teach the computer to perform algorithms to solve problems that are hard to process by a person. An example of this is the development of the calculator that performs operations with thousands of iterations in a few milliseconds.

Nevertheless, this kind of calculating algorithms is not enough, as there are other problems that could be easily solved if machines could process the resulting information of different medias. A

good example of this would be the facial recognition systems: identifying a person is an easy task for a human, but consulting thousands of different pictures to find one person is a process that would take various exhausting days. On the other hand, a machine could read different pictures and identify a specific person. To do so, it has to "see" the facial characteristics and compare the information gathered, but "seeing" with no eyes is not an easy task either.

Nowadays, one of the main areas of software development is the creation of systems that can simulate unreal scenes, the perception of which is easy-for-a-human, but difficult-for-a-machine [Shiffman, 2012]. Therefore, the technology must understand and learn non-linear aspects, process abstract information and predict or create scenarios according to previous conclusions: this is known as artificial intelligence.

During the second half of the twentieth century, a new technology based on the human brain called artificial neural networks was developed: systems designed to read signs and process the information, outputting a value according to the parameters established by results obtained previously. These are the most common systems applied to process information related to facial recognition, signal processing and prediction. Furthermore, this technology can be used for other purposes that are not related to solve specific problems, but as a different technique, for instance, to find patterns and to enhance the user experience of a social network, as the Style Transfer technique [Gatys *et al.* , 2015].

A.2 Convolutional Neural Networks

A.2.1 Neurons

Neural Neural networks arise from the attempt of modelling the biological neural system. "The nervous system is a complex collection of nerves and specialised cells known as neurons that transmit signals between different parts of the body" [Zimmermann, 2016]. This is basically a combination of sensors connected in an electrical wiring system. The problem of modelling it into an algorithm is to translate the power of learning and making decisions based on previous knowledge, just like teaching a child. In the human brain, neurons are connected to each other and are capable to read an input and transfer the information. But, if the decision turns out to be wrong, the human being is also capable of learning from the mistake.

As such, the artificial model created based on the neural system is called neural network, and it is composed by small units that mimic the performance of neurons (figure A.2). An artificial neuron is an elementary unit that receives input signals (e.g. x_0), these interact by a multiplicative factor (e.g. $w_0 \cdot x_0$) based on one value representative of weight (e.g. w_0), and produces an output signal along its axon. The intention of this algorithm is that the weights (e.g. w_0) are learned, and so with each operation these values can be readjusted. In the model in figure (A.3), the dendrites carry the signal to the cell body where they all get summed. If the final sum is above a certain threshold, the neuron sends a spike along its axon. The activation function (f) represents the frequency of the spikes along the axon. In other words, each neuron performs a dot product with

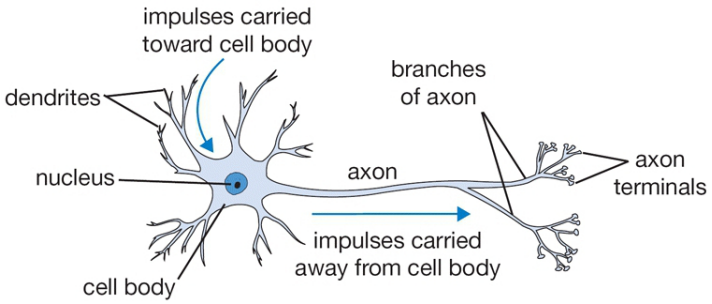

Figure A.2: Illustration of a Neuron [Karpathy, 2017b]

the input and its weights, adds the bias and applies the activation function to produce the output, which may serve as an input to the next neuron.[Karpathy, 2017b]

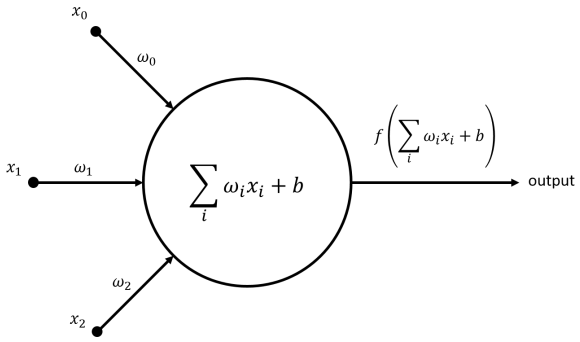

Figure A.3: Mathematical Model of a Neuron

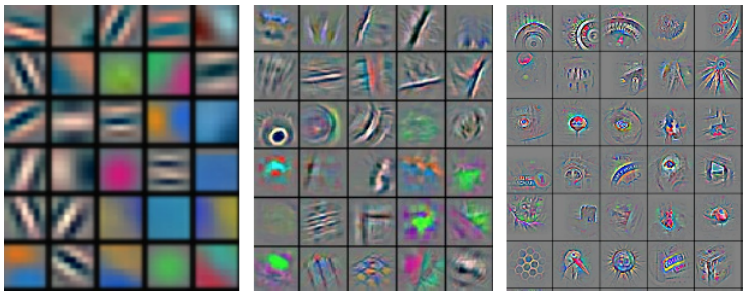

Figure A.4: Examples of features searched by the kernels

A.2.2 Layers

The network itself is a combination of different layers[1]: convolutional layers, rectified linear unit (ReLU) layers, pooling layers and fully connected layers. As the name suggests, the convolutional

[1]Hidden layers: the layers that have as input the outputs of other layers, and that produce outputs as inputs for following layers

layers read the input information and sum the values obtained by the comparison between a kernel and the input, as shown in figure A.5. The first layer of a CNN is always a convolutional layer. This layer applies a method that could be described as a small filter sliding over the input image. A filter (also known as a kernel, shown as examples in Figure A.4) is an array of numbers and the region that it is analysing is called receptive field. The input of the system is an image, which is also an array of numbers, with a size $L \times W \times 3$ (length, weight and colour code - e.g. RGB). The depth of the filter and of the input image must be the same, i.e., if the image has 3 levels for colour definition, then the kernel must have them too, and if the filter is of size $a \times b \times 3$, the receptive filter will have the same size exactly. Considering one filter of the first layer, the resulting output after the convolution operation will be an activation map (or feature map), and its size depends on the padding of the input. This is because the filter will only fit where values are defined, leaving the borders out of the candidate position for the filter. For instance, if the input is of size 32x32x3 and the kernel with dimensions 5x5x3, this filter will do 75 multiplications for each position, it will slide along the whole picture making a total of $5 \times 5 \times 3 \times 28 \times 28$ multiplications, having 784 values on the output, the feature map, with size $28 \times 28 \times 1$.

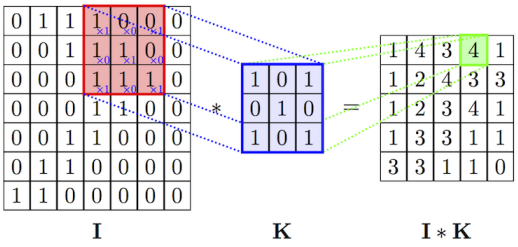

Figure A.5: Illustration of the convolutional process, [Buitendijk, 2003]

A possible method for handling borders is to assume that there are no values to compute the borders. Other options would be padding the image with zeros, padding the image with first and last values (image border extension) or repeat the image cyclically (resembling a mosaic). For the purpose of the project described in this dissertation, it is always considered some padding of zeros in the borders in order prevent loosing the information contained in the limits of the input image. As the filter is sliding, or convolving, around the image, it is multiplying the values in the filter with the original pixel values of the image, which means it is computing element wise multiplications, as shown in the equation (A.1) [2]. These multiplications are all summed up resulting in a single number, the value of which represents a single position of the filter. So the process is repeated for all positions in the input volume. Every unique location will produce a number and after sliding the filter over all possibilities, the result will be an array of numbers, which composes the feature

[2]This equation applies to a specific position in the input image, that is of size (x, y), multiplied by the filter, that is of size $(2N+1, 2N+1)$, and so, it has equal length sizes, and its centre is in (0,0).

map.

$$F \circledast I_{x,y} = \sum_{j=-N}^{N} \cdot \sum_{i=-N}^{N} (F_{i,j} \cdot I_{x-i,y-j}) \tag{A.1}$$

But one layer can have more than one filter and this would change the output size. For one kernel, the size in the example given above would be $28 \times 28 \times 1$, whereas with two kernels, there would be another feature map, so the output size would be 28x28x2, so there will be one feature map per kernel, increasing the output z dimension with the number of filters. Each of these filters can be viewed as feature identifiers, which on the first layer are as simple as straight edges, colours or curves, because they have to represent the basic characteristics that all images have in common. On the first layer, the filters convolve around the input image and activate, i.e., compute high levels, when the specific feature it is looking for is in the input volume at a certain position. The following layer of the system would have as inputs the feature maps of the previous layer. The higher the convolutional layer is, the less detail it will try to read. This means that if the system had as a goal to recognise a car in an input picture, the first layer would try to find small features, the second would be looking for more specific forms, like rims and windows, and so on until the higher levels, that would be searching for the the main form of a car, with less concern on the details.

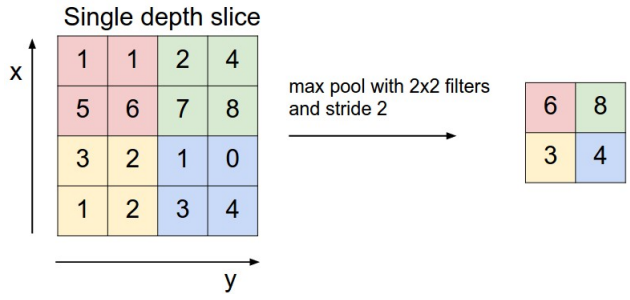

Figure A.6: Illustration of the max-pooling process, [Karpathy, 2017a]

For a system of object recognition, the output would be an answer: if the input image is a car or if not. But in the system of this Style Transfer project, there were only used the feature maps taken from a model trained on object recognition.

After a convolutional layer usually comes a ReLU layer, in order to rectify negative values and guarantee the system's stability. So in this layer, the positive values obtained in the feature maps are kept, but the negative values are replaced by zero. The pooling layers (figure [**?**]) are commonly used, as they are the ones responsible for the downsampling. So, having a feature map of $28 \times 28 \times 1$, the pooling layer would receive this and shrink it to an array of numbers with half of that size, if it used a window of 2x2. In this case, analysing a group of 4 pixels, it could apply an operation of max-pooling (taking the higher value of this group) or an average operation (considering the 4 values). Lastly, the fully connected layers are very used for making a decision

when aimed to an object recognition problem, but does not apply in the case of the Style Transfer technique.

A.3 Caffe as the Framework

When creating a Deep Learning Model, a framework can come in hand to facilitate its implementation, both to define a network and to train it for one defined purpose. There are several frameworks available, and they are distinguishable, for instance, considering if they are open source, what are the compatible languages for the creation of the model and the speed performance.

By the analysis of the possible frameworks, it is possible to conclude that Caffe [Jia *et al.* , 2014] is a suitable framework for the work described in this dissertation. Caffe supports various layers such as convolution, fully connected and pooling layers [Bahrampour *et al.* , 2015]. It is one of the most popular toolkit within the computer vision community [zer0n, 2016], despite the drawbacks in its installation. Caffe is open source, widely developed and fast [Bahrampour *et al.* , 2015].

In its foundation, Caffe defines a net layer-by-layer and because it is intended the information to flow forward and backwards, Caffe stores, communicates, and manipulates the information as blobs, which is the standard array and unified memory interface for the framework. [Jia & Shelhamer, 2014]

Appendix B

Deep Learning Models

Imagenet [Deng *et al.* , 2009] is one of the largest high-quality image datasets in the world. Imagenet has been hosting an annual challenge where research teams present solutions to image classification and other tasks by training on the ImageNet dataset. Concerning this, the Oxford University's VGG presented two network architectures [Simonyan & Zisserman, 2014]: VGG-16, a 16-layer convolutional Neural Network, and VGG-19, a 19-layer Convolutional Neural Network. Here the graphic representations of the models VGG-16 [Simonyan & Zisserman, 2015a], in figure B.1, and VGG-19 [Simonyan & Zisserman, 2015b], in figure B.2, can be found only to illustrate what is explained in previous chapters.

On the one hand, the VGG-16 model is organised in 13 convolutional layers, each of them followed by a ReLU layer. It is composed in 5 main stages, the first and second with 2 repetitions of the convolution operation and the three last stages with 3 repetitions. In the end of each of these stages there is a pooling layer for downsampling, making a total of 5 pooling layers.

On the other hand, VGG-19 is organised in 16 convolutional layers, each of them followed by a ReLU layer. It is also composed in 5 main stages, the first and second with 2 repetitions of the convolution operation and the three last stages with 4 repetitions. In the end of each of these stages there is a pooling layer for downsampling, making a total of 5 pooling layers.

Also, the ImageNet Large-Scale Visual Recognition Challenge 2014 had at first place for classification and detection the GoogLeNet [Szegedy *et al.* , 2014], designed to have a better occupation of computational resources along the network, but with a deeper and larger model. Its architecture is shown in Figure B.3, with 9 inception modules, which basically means that there is a network inside a network, inside a network and so on, not having a single path from input to output.

Comparing the three models, although VGG-19 comes as an improvement on object recognition tasks, it uses a high amount of resources and VGG-16 seems to work better with finer details. GoogLeNet is much faster so in this dissertation, it was very useful to produce fast results, but both VGG models are more accurate and produce more visually appealing results.

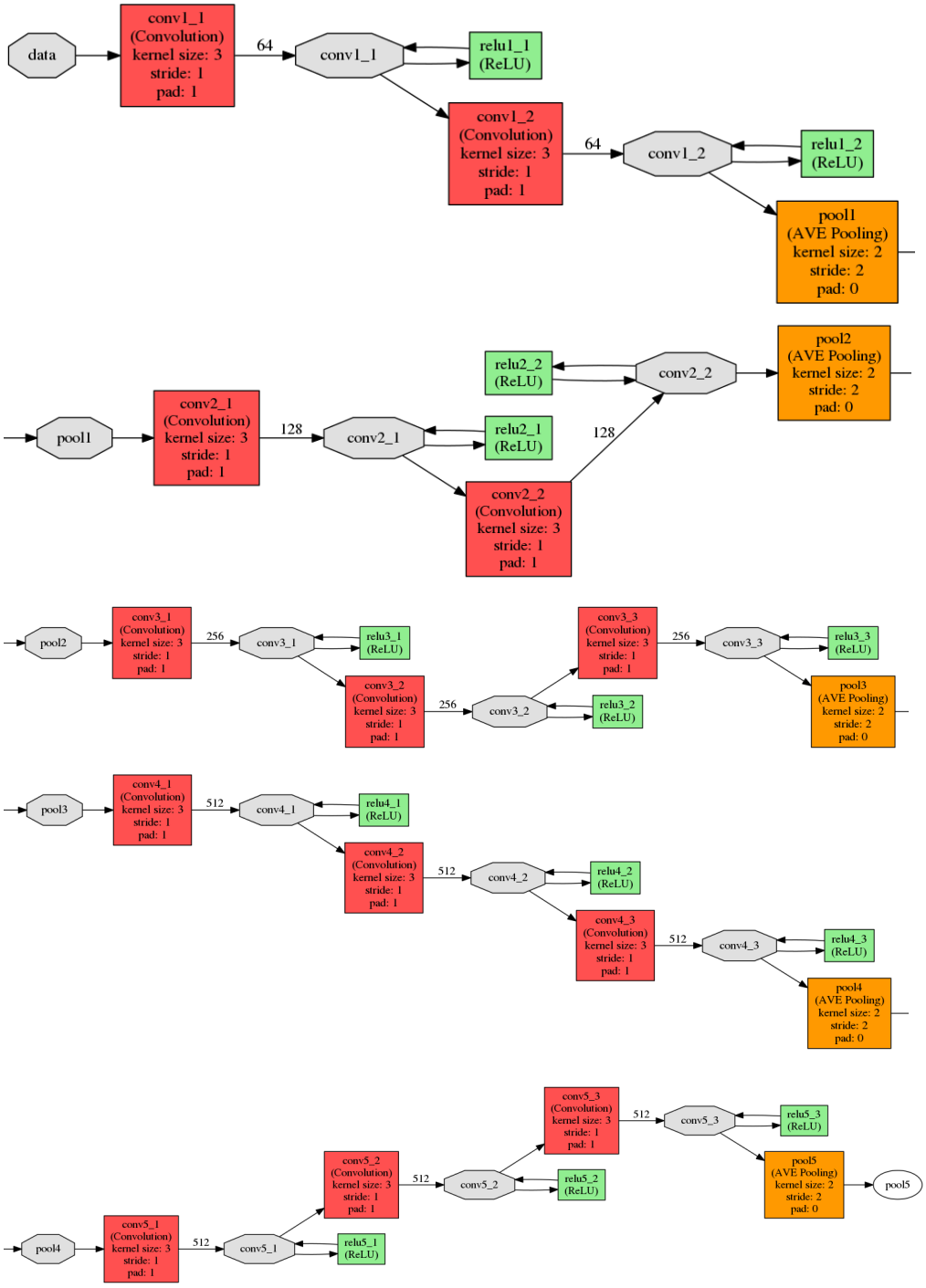

Figure B.1: VGG-16, in [Simonyan & Zisserman, 2015a]

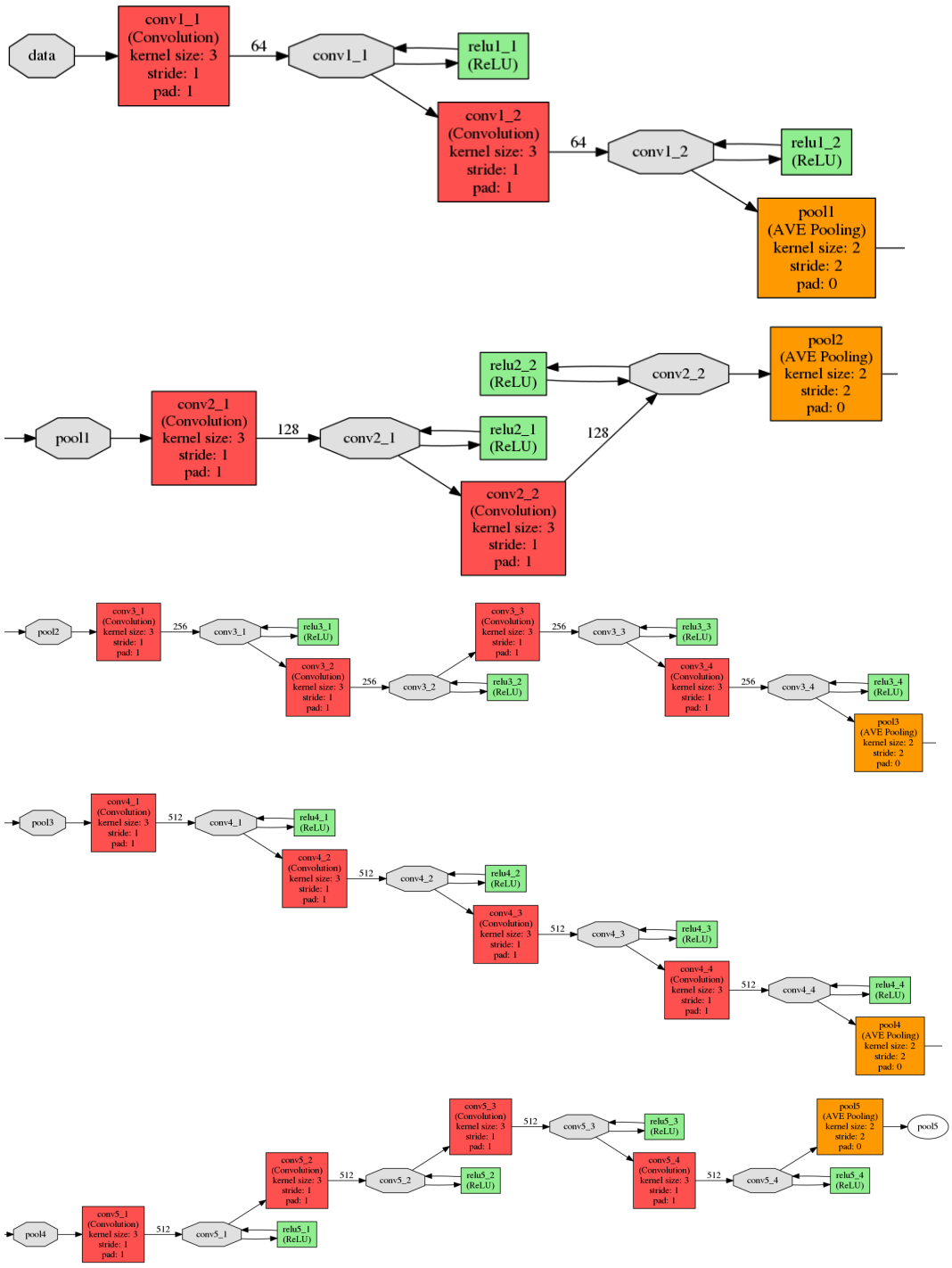

Figure B.2: VGG-19, in [Simonyan & Zisserman, 2015b]

Figure B.3: GoogLeNet (part 1), in [Szegedy *et al.* , 2014]

Figure B.4: GoogLeNet (part 2), in [Szegedy *et al.* , 2014]

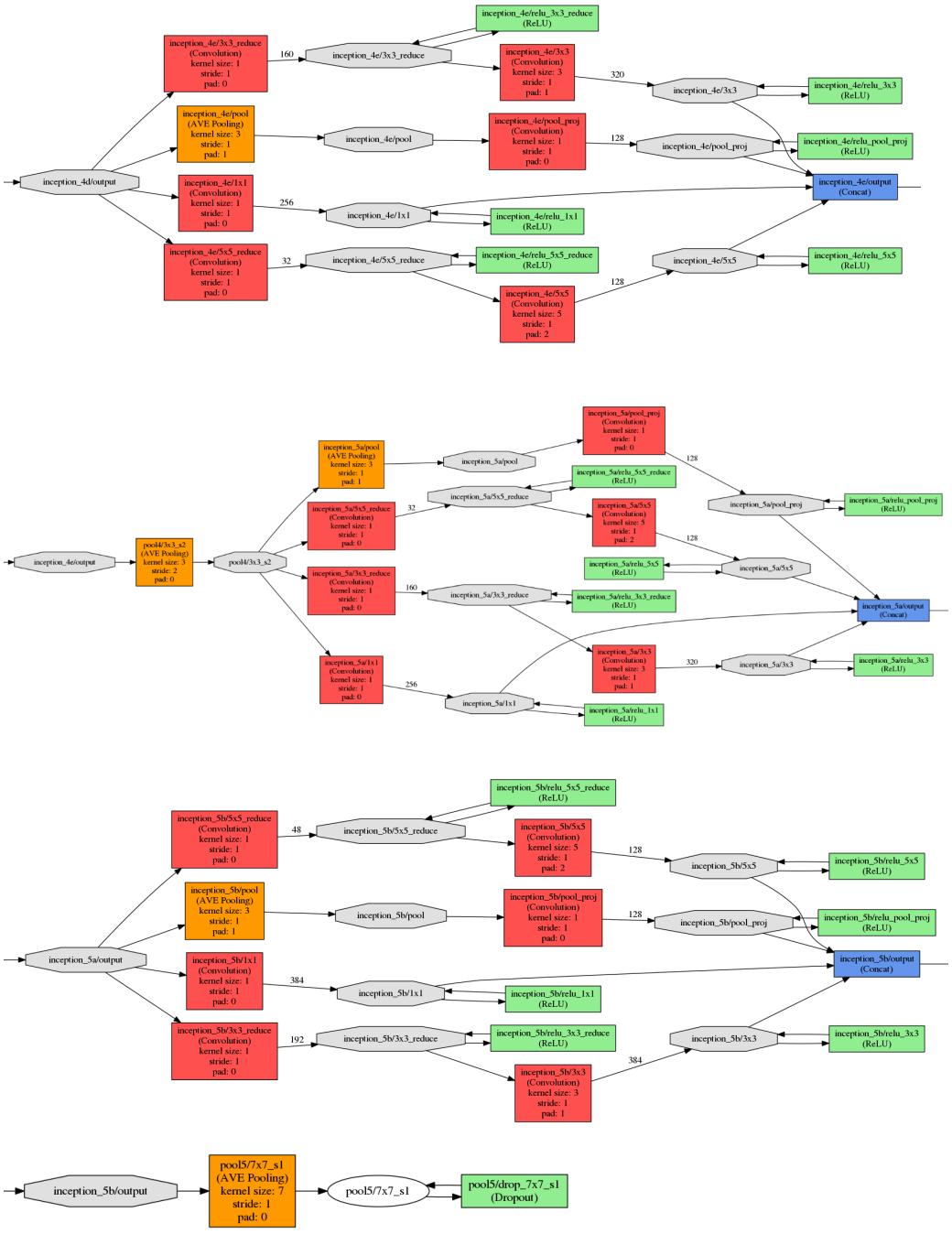

Figure B.5: GoogLeNet (part 3), in [Szegedy *et al.* , 2014]

Appendix C

Images

In this appendix, the images chosen for the style appliances, either for 2D or for 3D style transference, are presented in a list and properly labelled. Also, some content images referred in the text are also presented in this appendix.

Figure C.1: Les Demoiselles D'Avignon Picasso [1907]

Figure C.2: Girl with a Pearl Earring Vermeer [1665]

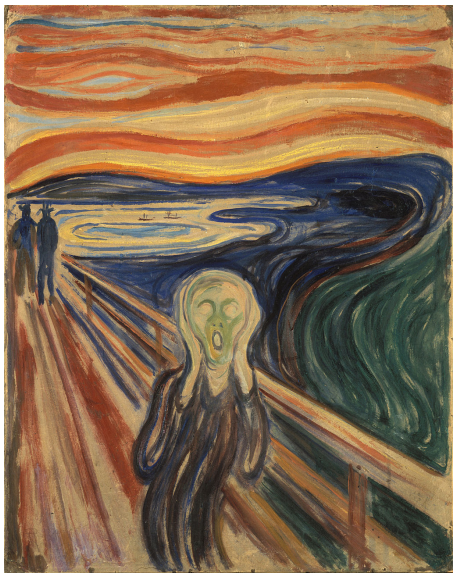

Figure C.3: The Scream, by Edvard Munch [Munch, 1910]

Figure C.4: Detail of the Goethe statue, Berlin

Figure C.5: Segment of the Berlin Wall - Birgit Kinder

Figure C.6: Guernica, by Pablo Picasso [Picasso, 1937]

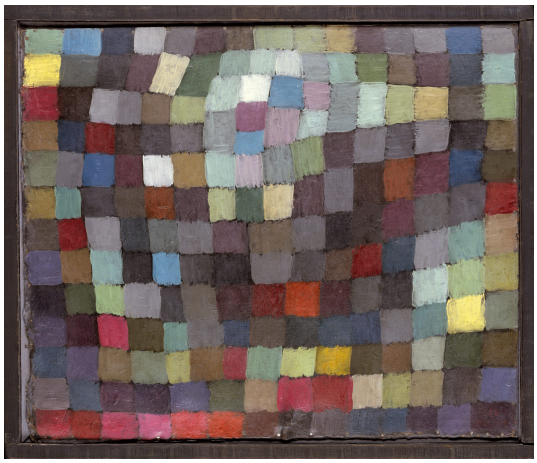

Figure C.7: May Picture, by Paul Klee [Klee, 1925]

Figure C.8: Vincent's Bedroom In Arles, by V. Van Gogh [Gogh, 1889]

Figure C.9: Unter den Linden, by Max Slevogt [Slevogt, 1913]

Figure C.10: Berlin Marchbrücke's view

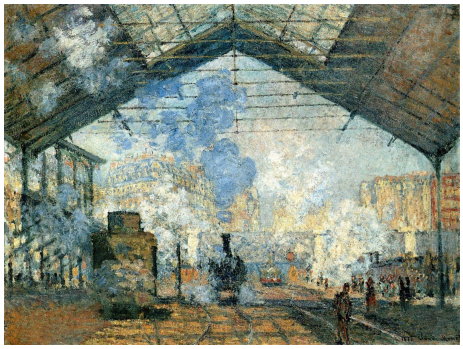

Figure C.11: Gare Saint-Lazare, by Claude Monet [Monet, 1877]

Figure C.12: Starry Night, by V. Van Gogh

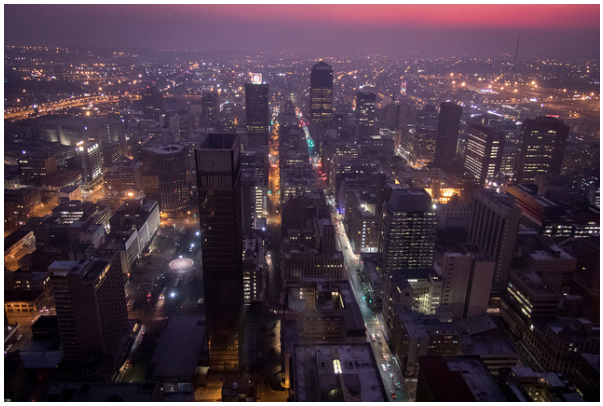

Figure C.13: Johannesburg

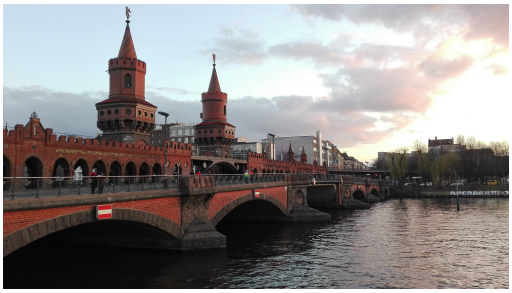

Figure C.14: Oberbaumbrücke, Berlin

Figure C.15: Berlin Cathedral

Figure C.16: Girl With Mandolin, by P. Picasso

Figure C.17: Goethe Statue, Berlin

References

Andy Grundberg, Helmut Erich Robert Gernsheim, Beaumont Newhall Naomi Rosenblum. 2017. *History of Photography*. https://www.britannica.com/technology/photography#toc252838.

Artindustri, Group. 2013. *Art Movements*. http://www.artmovements.co.uk/.

Artomatix. 2014. https://www.artomatix.com/.

Bahrampour, Soheil, Ramakrishnan, Naveen, Schott, Lukas, & Shah, Mohak. 2015. Comparative Study of Caffe, Neon, Theano, and Torch for Deep Learning. *CoRR*, **abs/1511.06435**.

Bear, Buddy. 2001. *Berlin Buddy Bear*. https://shop.buddy-baer.com/.

Braques, George. 1911. *The Portuguese*.

Buitendijk, Japp. 2003. *Girl with a pearl earring*. http://www.imdb.com/title/tt0335119/mediaviewer/rm3626997760.

Burt, P., & Adelson, E. 1983. The Laplacian Pyramid as a Compact Image Code. *IEEE Transactions on Communications*, **31**(4), 532–540.

Chen, Guangming, Li, Guiqing, Nie, Yongwei, Xian, Chuhua, & Mao, Aihua. 2016. Stylistic indoor colour design via bayesian network. *Computers & Graphics*, **60**, 34–45.

Cignoni, Paolo, Corsini, Massimiliano, & Ranzuglia, Guido. 2008. MeshLab: an Open-Source 3D Mesh Processing System. *ERCIM News*, April, 45–46.

Dahl, George E, Yu, Dong, Deng, Li, & Acero, Alex. 2012. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, **20**(1), 30–42.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. 2009. ImageNet: A Large-Scale Hierarchical Image Database. *In: CVPR09*.

Dromey, Trish. 2017. Artomatix is revolutionising the world of 3D content.

Dumoulin, Vincent, Shlens, Jonathon, & Kudlur, Manjunath. 2016. A Learned Representation For Artistic Style. *CoRR*, **abs/1610.07629**.

Efros, Alexei A, & Freeman, William T. 2001. Image quilting for texture synthesis and transfer. *Pages 341–346 of: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM.

Fišer, Jakub, Jamriška, Ondřej, Lukáč, Michal, Shechtman, Eli, Asente, Paul, Lu, Jingwan, & Sýkora, Daniel. 2016. StyLit: Illumination-Guided Example-Based Stylization of 3D Renderings. *ACM Transactions on Graphics*, **35**(4).

Foley, James D., Phillips, Richard L., Hughes, John F., Dam, Andries van, & Feiner, Steven K. 1994. *Introduction to Computer Graphics*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Fzliu. 2015. *Style Transfer*. https://github.com/fzliu/style-transfer.

Gatys, Leon A., Ecker, Alexander S., & Bethge, Matthias. 2015. A Neural Algorithm of Artistic Style. *CoRR*, **abs/1508.06576**.

Gatys, Leon A., Ecker, Alexander S., Bethge, Matthias, Hertzmann, Aaron, & Shechtman, Eli. 2016. Controlling Perceptual Factors in Neural Style Transfer. *CoRR*, **abs/1611.07865**.

Geng, W. 2011. *The Algorithms and Principles of Non-photorealistic Graphics: Artistic Rendering and Cartoon Animation*. Advanced Topics in Science and Technology in China. Springer Berlin Heidelberg.

Gogh, V. Van. 1889. *Vincent's Bedroom In Arles*. https://www.wikiart.org/en/vincent-van-gogh/vincent-s-bedroom-in-arles-1889-1.

Haeberli, Paul. 1990. Paint by numbers: Abstract image representations. *Pages 207–214 of: ACM SIGGRAPH computer graphics*, vol. 24. ACM.

Han, Zhizhong, Liu, Zhenbao, Han, Junwei, & Bu, Shuhui. 2015. 3D shape creation by style transfer. *The Visual Computer*, **31**(9), 1147–1161.

Hearn, D., & Baker, M.P. 1994. *Computer Graphics*. Prentice-Hall.

Hertzmann, Aaron, Jacobs, Charles E., Oliver, Nuria, Curless, Brian, & Salesin, David H. 2001. Image Analogies. *Pages 327–340 of: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '01. New York, NY, USA: ACM.

History. 2010. The Invention of the Internet. *History.com*.

Huang, Xun, & Belongie, Serge J. 2017. Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization. *CoRR*, **abs/1703.06868**.

Ikuta, Hikaru, Ogaki, Keisuke, & Odagiri, Yuri. 2016. Blending Texture Features from Multiple Reference Images for Style Transfer. *Pages 15:1–15:4 of: SIGGRAPH ASIA 2016 Technical Briefs*. SA '16. New York, NY, USA: ACM.

Jana, Reena. *Adobe survey: Creativity is important for career success*.

Jia, Yangqing, & Shelhamer, Evan. 2014. *Caffe tutorial*. http://caffe.berkeleyvision.org/tutorial/.

Jia, Yangqing, Shelhamer, Evan, Donahue, Jeff, Karayev, Sergey, Long, Jonathan, Girshick, Ross, Guadarrama, Sergio, & Darrell, Trevor. 2014. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093*.

Johnson, Justin, Alahi, Alexandre, & Fei-Fei, Li. 2016a. Perceptual losses for real-time style transfer and super-resolution. *Pages 694–711 of: European Conference on Computer Vision*. Springer.

Johnson, Justin, Alahi, Alexandre, & Li, Fei-Fei. 2016b. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. *CoRR*, **abs/1603.08155**.

Joshi, Bhautik J., Stewart, Kristen, & Shapiro, David. 2017. Bringing Impressionism to Life with Neural Style Transfer in Come Swim. *CoRR*, **abs/1701.04928**.

Karpathy, Andrej. 2016. *CS231n Neural Networks*. https://www.youtube.com/watch?v=gYpoJMlgyXA.

Karpathy, Andrej. 2017a. *CS231n Convolutional Networks*. http://cs231n.github.io/convolutional-networks/.

Karpathy, Andrej. 2017b. *CS231n Convolutional Neural Networks for Visual Recognition*. http://cs231n.github.io/neural-networks-1/.

Karpathy, Andrej. 2017c. *CS231n Transfer Learning*. http://cs231n.github.io/transfer-learning/.

Klee, Paul. 1925. *May Picture*. http://www.metmuseum.org/toah/images/hb/hb_1984.315.42.jpg.

Krizhevsky, Alex, Sutskever, Ilya, & Hinton, Geoffrey E. 2012. Imagenet classification with deep convolutional neural networks. *Pages 1097–1105 of: Advances in neural information processing systems*.

Kyprianidis, Jan Eric, Collomosse, John, Wang, Tinghuai, & Isenberg, Tobias. 2013. State of the "Art”: A Taxonomy of Artistic Stylization Techniques for Images and Video. *IEEE Transactions on Visualization and Computer Graphics*, **19**(5), 866–885.

Larochelle, Hugo. 2016. *The deep end of deep learning, TEDxBoston*. Twitter, https://www.youtube.com/watch?v=dz_jeuWx3j0.

Li, Wilmot, Agrawala, Maneesh, Curless, Brian, & Salesin, David. 2008. Automated Generation of Interactive 3D Exploded View Diagrams. *ACM Trans. Graph.*, **27**(3), 101:1–101:7.

Litwinowicz, Peter. 1997. Processing images and video for an impressionist effect. *Pages 407–414 of: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co.

Lu, Jingwan, Sander, Pedro V., & Finkelstein, Adam. 2010. Interactive Painterly Stylization of Images, Videos and 3D Animations. *Pages 127–134 of: Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D '10. New York, NY, USA: ACM.

Luan, Fujun, Paris, Sylvain, Shechtman, Eli, & Bala, Kavita. 2017. Deep Photo Style Transfer. *CoRR*, **abs/1703.07511**.

Ma, Chongyang, Huang, Haibin, Sheffer, Alla, Kalogerakis, Evangelos, & Wang, Rui. 2014. Analogy-driven 3D style transfer. *Pages 175–184 of: Computer Graphics Forum*, vol. 33. Wiley Online Library.

Moiseenkov, Alexey. 2016. *Prisma*. https://prisma-ai.com/.

Monet, Claude. 1877. *Gare Saint-Lazare*. https://upload.wikimedia.org/wikipedia/commons/1/1f/La_Gare_Saint-Lazare.jpg.

Munch, Edvard. 1910. *The Scream*. https://upload.wikimedia.org/wikipedia/commons/8/86/Edvard_Munch_-_The_Scream_-_Google_Art_Project.jpg.

Munoz, Andres. 2014. Machine Learning and Optimization.

Narayanan, Harish. 2017. *Convolutional neural networks for artistic style transfer*. https://harishnarayanan.org/writing/artistic-style-transfer/.

Neumann, László, & Neumann, Attila. 2005. Color style transfer techniques using hue, lightness and saturation histogram matching. *Pages 111–122 of: Computational Aesthetics*.

Nguyen, Chuong H, Ritschel, Tobias, Myszkowski, Karol, Eisemann, Elmar, & Seidel, Hans-Peter. 2012. 3D material style transfer. *Pages 431–438 of: Computer Graphics Forum*, vol. 31. Wiley Online Library.

Nielsen, Michael. 2017. *Neural Networks and Deep Learning*.

Novak, Roman, & Nikulin, Yaroslav. 2016. Improving the neural algorithm of artistic style. *arXiv preprint arXiv:1605.04603*.

Parade, Cow. 1999. *Cow Parade*. http://www.cowparade.com.

Picasso, Pablo. 1907. *Les Demoiselles d'Avignon*. http://colourlex.com/wp-content/uploads/2015/02/Les$_D$*emoiselles*$_d$*Avignon* − *wiki* − 600*b.jpg*.

Picasso, Pablo. 1937. *Guernica*. https://i.imgbox.com/hVyqyKDz.jpg.

Pitie, Francois, Kokaram, Anil C, & Dahyot, Rozenn. 2005. N-dimensional probability density function transfer and its application to color transfer. *Pages 1434–1439 of: Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 2. IEEE.

Pouli, Tania, & Reinhard, Erik. 2011. Progressive color transfer for images of arbitrary dynamic range. *Computers & Graphics*, **35**(1), 67–80.

Reinhard, E., Adhikhmin, M., Gooch, B., & Shirley, P. 2001. Color transfer between images. *IEEE Computer Graphics and Applications*, **21**(5), 34–41.

Ruder, Manuel, Dosovitskiy, Alexey, & Brox, Thomas. 2016a. Artistic style transfer for videos. *CoRR*, **abs/1604.08610**.

Ruder, Manuel, Dosovitskiy, Alexey, & Brox, Thomas. 2016b. Artistic style transfer for videos. *Pages 26–36 of: German Conference on Pattern Recognition*. Springer.

Schroepfer, Mike. 2016a. *Accelerating innovation and powering new experiences with A.I.* https://newsroom.fb.com/news/2016/11/accelerating-innovation-and-powering-new-experiences-with-ai/.

Schroepfer, Mike. 2016b. *Ten years from now*. Web Summit 2016 Lisbon, Facebook CTO https://www.youtube.com/watch?v=9avdHu40Enw&t=901s.

Selim, Ahmed, Elgharib, Mohamed, & Doyle, Linda. 2016. Painting style transfer for head portraits using convolutional neural networks. *ACM Transactions on Graphics (TOG)*, **35**(4), 129.

Shiffman, Daniel. 2012. *The Nature of Code*.

Simonyan, Karen, & Zisserman, Andrew. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, **abs/1409.1556**.

Simonyan, Karen, & Zisserman, Andrew. 2015a. *16-layer model*.

Simonyan, Karen, & Zisserman, Andrew. 2015b. *19-layer model*.

Slevogt, Max. 1913. *Unter den Linden*. https://uploads7.wikiart.org/images/max-slevogt/unter-den-linden.jpg.

Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott E., Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, & Rabinovich, Andrew. 2014. Going Deeper with Convolutions. *CoRR*, **abs/1409.4842**.

theartstory. 2009. *Definition of Modern Art*. http://www.theartstory.org/definition-modern-art.htm.

Titcomb, James. 2015. Google unleashes machine dreaming software on the public, nightmarish images flood the internet. *The Telegraph*.

Tomasco, Steve. *IBM 2010 Global CEO Study: Creativity Selected as Most Crucial Factor for Future Success*.

Treavett, SMF, & Chen, M. 1997. Statistical techniques for the automated synthesis of non-photorealistic images. *Pages 201–210 of: Proc. 15th Eurographics UK Conference*.

Tyka, Mike. 2015a. *The art of neural networks, TEDxTUM*. Google, https://www.youtube.com/watch?v=0qVOUD76JOg.

Tyka, Mike. 2015b. *Experiments with style transfer*. Google, https://mtyka.github.io/.

Tyka, Mike. 2015c. *Inceptionism*. Google, http://www.miketyka.com/.

Vaynerchuk, Gary. 2016. *Keynote*. Web Summit 2016 Lisbon, VaynerMedia CEO https://www.youtube.com/watch?v=8BDHQBXm_cA.

Vermeer, Johannes. 1665. *Girl with a pearl earring*. http://www.artble.com/imgs/8/8/c/218507/girl_with_a_pearl_earring.jpg.

Vigoda, Benjamin. 2016. *When machines have ideas, TEDxBoston*. Founder, CEO/CTO of Gamalon Machine Intelligence, https://www.youtube.com/watch?v=PCs3vsoMZfY.

Wilmot, Pierre, Risser, Eric, & Barnes, Connelly. 2017. Stable and Controllable Neural Texture Synthesis and Style Transfer Using Histogram Losses. *CoRR*, **abs/1701.08893**.

Xiao, Xuezhong, & Ma, Lizhuang. 2009. Gradient-Preserving Color Transfer. *Pages 1879–1886 of: Computer Graphics Forum*, vol. 28. Wiley Online Library.

Yongcheng Jing, Yezhou Yang, Zunlei Feng Jingwen Ye, & Song, Mingli. 2017. Neural Style Transfer: A Review. *CoRR*, **abs/1705.04058**.

Zeng, Kun, Zhao, Mingtian, Xiong, Caiming, & Zhu, Song-Chun. 2009. From image parsing to painterly rendering. *ACM Trans. Graph*, **29**(1), 2.

zer0n. 2016. *Evaluation of Deep Learning Frameworks*. https://github.com/zer0n/deepframeworks.

Zimmermann, Kim Ann. 2016. *Nervous System: Facts, Function Diseases*. http://www.livescience.com/22665-nervous-system.html.